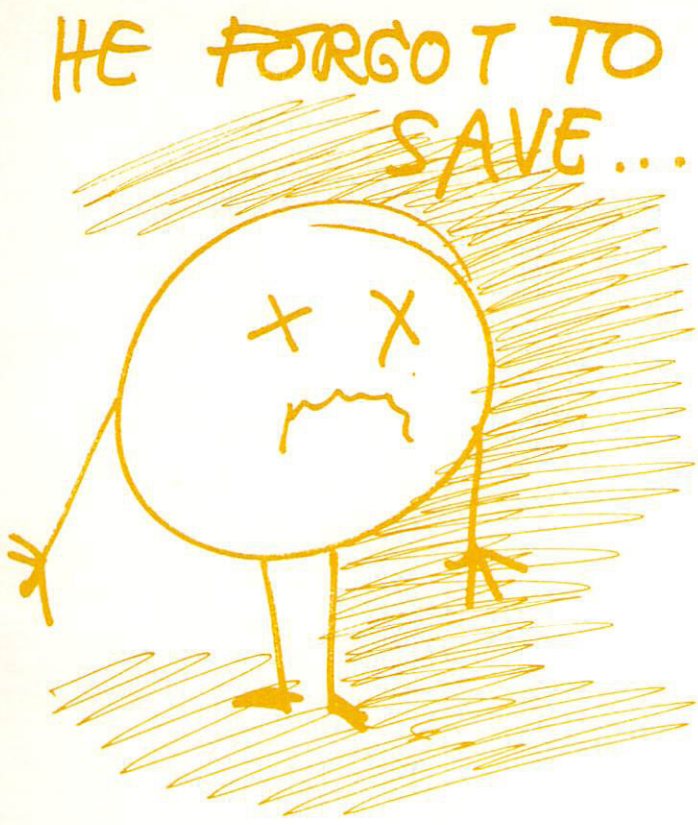



# COMAL Today

# 6

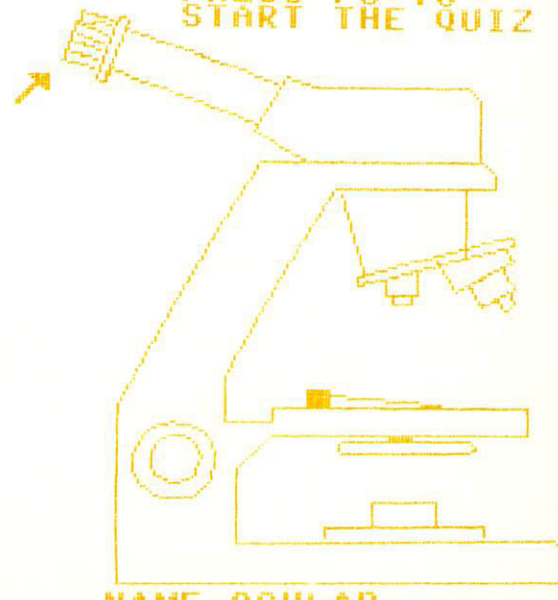
## CAPTAIN COMAL™



MICROSCOPE QUIZ
BY MIKE DRYJA

PRESS F1 TO GO ON TO NEXT PART

PRESS F3 TO START THE QUIZ



NAME!OCULAR

This program is on TODAY DISK #6

# TABLE OF CONTENTS

COMAL TODAY #6 - March 8, 1985
Publisher: COMAL Users Group U.S.A. Ltd.
5501 Groveland Ter, Madison, WI 53716
Editor: Len Lindsay // (608) 222-4432
Captain COMAL artwork: Frank Hejndorf



# HOW TO TYPE IN PROGRAMS

Line numbers are irrelevant to a running COMAL program. COMAL only provides line numbers for our benefit in editing the program. Thus most magazines do not use line numbers when listing a COMAL PROGRAM. It is up to **YOU** to provide the line numbers. But of course, **COMAL can do it for you** quite easily. Just follow these steps to type in a COMAL program listing:

1) Enter command: NEW
2) Enter command: AUTO
3) Type in the program
4) Hit RETURN key twice when done (0.14) or Hit STOP key (2.0)

Remember - use unshifted letters thoughout entering the program. If letters are capitalized in the listing it does not mean use the SHIFT with those letters. They are capitalized merely for a pleasant, easy to read, look. The only place to use SHIFTED letters is inside quotes. Also, you don't have to type leading spaces in a line. They are listed only to emphasize structures. You do have to type a space between COMAL words in the program.

**LONG PROGRAM LINES:** We are continuing to print COMAL TODAY with two columns per page, printed in elite (12 pitch) with 40 characters maximum per line. This makes it easiest to read. However, some program listings have program lines that extend beyond the 40 character limit. We decided to list these lines in the same manner that COMAL uses when listing long lines on a 40 column screen. We simply break the line right at the 40 character spot, and continue it on the next line, indenting it properly to keep the program structures obvious. These are called wrap lines. To draw your attention to these continued lines we add a **//wrap line** comment to the end of the line. Whenever you see this make sure you type both lines as one continuous program line! The following example includes a line with more than 40 characters that we must list on two lines, but you must type in as one long program line:

```
IF CURRENT'NAME$<>"STOP" THEN PRINT'LABE
L(CURRENT'NAME$,PHONE$) //wrap line
```

If you type in this long program line as two shorter program lines, COMAL will not object (although sometimes it will)! But, the program will not work unless it is entered as one long line. The procedure name PRINT'LABEL is split onto two lines in the listing, but the //wrap line draws your attention to this fact.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

COMAL TODAY welcomes contributions of articles, manuscripts and programs which would be of interest to readers. All manuscripts and articles sent to COMAL TODAY will be treated as unconditionally assigned for publication and copyright purposes to COMAL Users Group, U.S.A., Limited and is subject to the Editor's unrestricted right to edit and to comment editorially. Programs developed and submitted by authors remain their property, with the exception that COMAL Users Group, U.S.A., Limited reserves the right to reprint the materials, based on that published in COMAL TODAY, in future publications. There will be no remuneration for any contributed manuscripts, articles or programs. These terms may be varied only upon the prior written agreement of the Editor and COMAL Users Group, U.S.A., Limited. Interested authors should contact the Editor for further information. All articles and programs should be sent to COMAL Users Group, U.S.A., Limited, 5501 Groveland Terrace, Madison, WI 53716-3251. Authors of articles, manuscripts and programs warrant that all materials submitted are original materials with full ownership rights resident in said authors. No portion of this magazine may be reproduced in any form without written permission from the publisher. Local Users Groups may reprint material from this issue if credit is given to COMAL TODAY and the author.  The opinions expressed in contributed articles are not necessarily those of COMAL Users Group, U.S.A., Limited. Although accuracy is a major objective, COMAL Users Group, U.S.A., Limited cannot assume liability for article/program errors.

# COMAL TODAY SUBSCRIBER SPECIALS

Here are this issues specials for subscribers only. If you subscribe now, you can take advantage of these specials at the same time. If you are a current subscriber, you **MUST** include your subscriber number with your order, or we will not honor the special prices. Your subscriber number is printed on the newsletter mailing label. New subscribers get these prices automatically and a number will be assigned with the order.

## ANY COMAL DISK ONLY $9.75

Now you can get any COMAL disk for only $9.75 - including our double sided disks. Or get our 19 disk boxed set for only $94.05.

## ANY MATCHING DISK ONLY $4.95

When you get one of the COMAL tutorial books, you can get it's matching disk at the same time for only $4.95.

## CAPTAIN COMAL BOOK SPECIALS

CAPTAIN COMAL GETS ORGANIZED, LIBRARY OF PROCEDURES AND FUNCTIONS, and GRAPHICS PRIMER book/ disk sets, usually $19.95, now only $14.95.

## C64 GRAPHICS WITH COMAL BOOK

The graphics reference book for COMAL 0.14 is finally available. The list price is $17.95 - but our special price to subscribers is only $14.95. Ask for COMMODORE 64 GRAPHICS WITH COMAL.

## COMAL HANDBOOK BY THE BOX

**USER GROUPS TAKE NOTE:** We have 6 unopened boxes full of the first edition of the COMAL HANDBOOK. There are 34 books in each 37 pound box. Buy the whole box for just $125 plus $20 shipping. These books are great for anyone using COMAL 0.14, only one page of notes (included) is needed. At just over $3 per book plus shipping how can your group go wrong? (Yes, that's less than the COMAL FROM A TO Z BOOK). A street address is required for UPS shipment. First come first served so act now.

## DISK SUBSCRIPTION ONLY $49.95

Subscribe to the TODAY disks now at a special price. These disks are double sided begining with TODAY DISK #6. Six disks (12 sides!) for only $49.95, and that includes the shipping! **EVEN MORE:** Extend your disk subscription beyond the usual 6 disks. Add onto your disk subscription as many disk issues as you wish for only $5 each. That is only about $2.50 per disk side! How can you refuse. Look at all the GOOD COMAL programs that you will get, all ready to run. And you can have your subscription start with any issue you wish, even issue #1. Take advantage of this offer while it lasts!

## NEWSLETTER SUBSCRIPTION

Subscribe or renew your subscription now, before the price goes up this summer. Only $14.95 for 6 issues. **EVEN MORE:** extend your newsletter subscription or renewal beyond the usual 6 issues. Add on as many issues as you want for only $2 each. Thus a 10 issue newsletter subscription would cost $14.95 (6 issues) plus $8 (4 added issues) or only $22.95. Cash in now before the subscription prices go up. Why worry about renewals in the future - get a 48 issue subscription now for only $98.95 (prepare for when COMAL TODAY goes monthly - at a LOW price now). All new subscriptions start with the current issue.

## FREE PLAYNET SETUP OFFER

COMAL ONLINE SUPPORT is now available via PlayNET (see page 46 for story). It normally costs $39.95 to setup your PlayNET account (only $2 per hour after that). Until May 30, 1985 we can set up your PlayNET account for FREE. We will send you the manual and complete PlayNET system disk FREE (you just pay the $2 shipping cost). To qualify for this offer you only need to do one of two things:

* Subscribe or renew a TODAY DISK subscription.
* Subscribe or renew a COMAL TODAY newsletter subscription for **10 or more** issues.

## COMAL NEEDS YOUR SUPPORT

COMAL is advancing fast. But to achieve the popularity it deserves we need your support now.

We have set up a COMAL section in the BBS part of PlayNET for multi user national online support. Plus every Thursday night, 10pm - midnight E.S.T. you will find Captain COMAL available on line in the COMAL room to answer questions and provide you with the latest COMAL information. We have also arranged with PlayNET to provide COMAL TODAY subscribers with a PlayNET account without the usual $39.95 set up fee (see page 1).

We have collected over 2,000 COMAL programs and have about 40 different COMAL disks. And now our TODAY disks will be double sided - like two disks for the price of one! And we lowered the price at the same time. Subscribe to 6 TODAY disks for only $49.95 and only $5 each disk after the first 6 (that is like only $2.50 per disk counting the double sides). We hope all of you will now be able to get the disks. We have over $10,000 invested in disks already duplicated just waiting for your order.

And eventually, everyone wants a couple COMAL books. So we try to keep them all in stock for you. But since we have a hard time with the publishers (we have had BEGINNING COMAL on order since about OCT 1984) we are now stocking the books by the HUNDREDS! WE have over 1000 copies of COMAL HANDBOOK in stock now, plus 500 FOUNDATIONS, 130 COMMODORE 64 GRAPHICS WITH COMAL, etc. We hope you will support our efforts in maintaining this huge inventory.

And now, FINALLY, the ROM version COMAL 2.0 Cartridges are IN STOCK. Yes, in stock for immediate delivery. And more are on order to meet your needs. If you want the BEST for your computer (and for you!) then you need the COMAL 2.0 Cartridge. As you can see from this issue, COMAL TODAY is prepared to support the COMAL 2.0 Cartridge all the way - without neglecting COMAL 0.14. Both versions are final versions, both will continue to be widely used, and both will be supported by COMAL TODAY.

COMAL is here to stay. It now has IBM's name on it. The IBM PC COMAL is due out this month from IBM DENMARK. And be assured that COMAL Users Group, U.S.A., Limited is here to stay too. We just invested in a 20 MEG removable hard disk drive to hold our subscriber records! We are trying our best here. We now have technical support available to answer your questions MON, WED, and FRI from 11am - 3pm CST. We also have programmers working on advanced COMAL programs. And the COMAL TODAY newsletter/ magazine staff is going to be growing. With added people we can put out the newsletter monthly we hope. We hope you like COMAL. We hope to have your support.

So, I'll be watching for your 48 issue COMAL TODAY subscription renewal, and the matching TODAY DISK subscription too. And take advantage of our books specials while you are at it. And most important - remember that you are important. Send in your notes, articles and programs today. Send them on disk - programs and articles in word processor files - preferably PaperClip or EasyScript - and we'll send you a COMAL disk back in return.

You are actively helping change the history of how people use their home computers. We're glad to have you with us. Hope you enjoy the ride.

**NOW YOU CAN CALL**
**TOLL FREE**
**24 HOURS A DAY**
**TO PLACE VISA/MC**
**CHARGE ORDERS**

**1-800-356-5324 EXT 1307**

**OR CALL DENISE HERE AT**
**OUR INFO LINE:**

**608-222-4432**

# GETTING STARTED WITH COMAL 2.0

or SOME CARTRIDGE QUESTIONS & ANSWERS
by Colin Thompson

No one involved with the release of the COMAL 2.0 cartridge imagined that a computing novice would buy the cartridge. It was assumed that experienced programmers would buy the first 1000 or so carts. It didn't turn out that way. In the week following the arrival of carts, I fielded more than 100 phone calls from new cart owners. Their questions ranged from "Should I plug in the cart first, or what?" to "How do I LOAD a program?".

For the benefit of those (few) of you that don't have my home phone number, (just kidding!), what follows here are the most common questions and my answers.

Q. What do I do first? I'm not a programmer.
A. Plug in the cart, long lip up, turn on the power and put cart demo disk #2 in the drive. Hold the SHIFT key down and at the same time, press the RUN/STOP key. Watch the screen and turn up the volume on your tv set. Enjoy. Read on.

Q. I ordered the cart months ago. I didn't learn COMAL 0.14 because I wanted to wait for the really powerful version to start with. I'm overwhelmed. Help.
A. You should now be aware of your mistake. COMAL 0.14 is a LEARNING LANGUAGE. Hint, Hint. If you don't have the background to pick up 2.0 on your own, GOTO COMAL 0.14 and learn that. There are many text books, reference manuals and an endless supply of programs to learn with.

Q. How do I LOAD a program?
A. To LOAD a program (regardless of which language you use), you must know what programs are on the diskette. To find out, type in CAT and press return. The disk's 'table of contents', called the directory will now start scrolling up your screen. You may pause the scrolling by pressing the SPACE BAR. Press it again and scrolling resumes. Press RUN/STOP to stop the directory. Now look at the list of programs. These are FILENAMES. (the names of files). Every FILENAME has a FILETYPE. (the type of file). Here is an example of what a directory looks like:

```
0  "demo disk #2    " pw 2a (DISK's NAME)

 >-- # of Blocks
 !       >-- Filename
 !       !                >-- Filetype
 V       V                V
--    -------------       ---
24    "all'at'once"       prg
20    "april'fool"        prg
14    "arabesque2"        prg
14    "bach'music"        prg
1     "bat.commands"      seq
14    "dat.bwv781"        seq
17    "font.mirror"       seq
2     "func.binary$"      seq
4     "lst.getnum"        seq
1     "shap.santa2"       seq
40    "hrg.hamburger"     seq
2  blocks free.         <-----BLOCKS FREE
```

Before you can LOAD a program, you must know what a program is. Look at the sample directory. The first four filenames have a PRG as the filetype. These four are programs that you can LOAD like this:

load "all'at'once" (return)

When the program LOADs, type in RUN and press return. That's one way to do it. Here is a different way:

run "all'at'once" (return)

This will LOAD the program and then RUN it automatically. There is another way also:

Type CAT (return).

When the directory is on the screen, move the cursor up to the filename you want to LOAD. In this example, put the cursor on the "2" in "24". This is the number of blocks for "all'at'once". Now press the f7 key. This will make the word "RUN" appear over the "24". The "PRG" following the filename will be erased and the program will LOAD and RUN automatically. This method is preferred - it eliminates typing errors and the dreaded "FILE NOT FOUND" error.

That's how to LOAD a program. Use the f7 key whenever you can. Look at the sample directory again. The remaining 7 filenames have a SEQ filetype. You cannot LOAD or RUN a SEQuential file. SEQ files

are special. Usually they hold data in a special format. This data is used by other programs. You can get some idea of what kind of data is held in the SEQ file by looking at the PREFIX of the filename. Here are some prefixes and their meanings:

HRG. = Hi Res Graphic picture.
FONT.= Font file. A set of characters.
DAT. = General purpose DATA file.
FUNC.= A FUNCtion, LISTed to disk.
PROC.= A PROCedure LISTed to disk.
SHAP.= Sprite images.
LST. = A COMAL 2.0 PRG LISTed to disk.
(you may ENTER this program)
TXT. = Text file from a wordprocessor.
BAT. = Batch file for SELECT INPUT use.

You can't LOAD or RUN any file that has a prefix, or is SEQuential. Now more questions.

Q. I really dislike having to unplug the cart every time I want to use BASIC, or load the directory of a BASIC disk. Isn't there an easier way?
A. Many folks didn't know that a COMAL disk and a BASIC disk look the same to the disk drive. When you want to look at a BASIC directory, use CAT. It works fine. When you want to use BASIC, type in BASIC and press return. From BASIC, you can go back to COMAL by typing SYS50000. Try it. You NEVER have to take out the cart.

Q. I bought the Cart so I could write a database program like SuperBase, only more powerful. I don't know how to write programs, yet. Where do I start.
A. You might be able to do it, but if you are like the rest of us mortals, be prepared to spend 2,000 hours of intense learning. Powerful databases are the last thing a programmer might attempt to write before he is committed to the California Home for the Permanently Unbalanced. Sell your computer and take up pyramid building.

Q. I want to write some musical scores. Is there any software for the cart yet?
A. There is a Music Editor/Player on the 2.0 Collection # 2. Cart Demo disks 3 and 4 have a different system, so there is much to choose from.

Q. The two disks that came with the cart were supposed to be DEMO disks. One of mine is a 'handbook' disk. The other is DEMO disk #2. What gives?
A. It's ok. The handbook disk is really DEMO disk #1. It has a lot of programs that explain how to use various keywords.

Q. Can I plug in another cartridge, like my favorite game, along with the COMAL Cart? Will they work together?
A. No. The COMAL Cart is really a whole new computer. When you plug it in, your C64 is now a COMAL computer that uses some of the C64's internal functions.

Q. Can I use the Buscard and a dual drive with the cart?
A. Yes. It's a good combination. The Buscard is transparent to COMAL.

Q. In COMAL 0.14, I couldn't print the disk's directory to the printer. Can I do it in 2.0?
A. Yes. Type SELECT "lp:" (return)
Type CAT (return). The directory will print in upper/lower case to the printer. When it stops, type SELECT"ds:" (return)

By now, you've probably caught on that any command you type in must be followed by pressing the RETURN key, so I won't mention it again. More questions:

Q. I'm a budding programmer. I can write simple programs in BASIC and COMAL 0.14. I want to see my 0.14 programs running under 2.0, but when I LOAD them, COMAL tells me the file is "NOT A COMAL PROGRAM". I know it's a COMAL program- I wrote it! What gives?
A. What the error message is telling you is that the program is not a 2.0 program. 2.0 and 0.14 store programs in different formats on the disk. You can run 0.14 programs in 2.0, but a little preparation is needed. Here's what to do. In 0.14, load the program, LIST it to disk. From 2.0, ENTER the listed file. Here's where the fun begins. The 0.14 listfile might ENTER without a hitch. If it does, save it to disk and then RUN it. (press f7). That would be unusual. What usually happens is this:
While the 0.14 file is coming in from the disk, the process stops and a line of code is displayed on the screen, along with a 2.0 error message. The cart has

found a problem with this line and wants you to fix it before it enters any more lines. Read the error message. The most common error is ""(" EXPECTED, NOT NUMERIC VARIABLE". 2.0 programs require a set of parentheses around parameters in graphics and sprite commands. Use the cursor keys and put in the needed parens. Then press return. If you can't figure out what COMAL wants, drag out the 2.0 booklet that came with the cart and read the appropriate section. If that fails, cheat. Put the cursor directly over the space following the line number. Type in a shifted one (exclamation point). Press return. That always works. You just REMed out the line. When the file is completely entered, SAVE it and then go back to see what you have to convert. Here's a brief list of syntax differences.

| COMAL 0.14 | COMAL 2.0 |
|---|---|
| settext | textscreen |
| setgraphic 0 | graphicscreen (0) |
| clear | clearscreen |
| pencolor 5 | pencolor (5) |
| plottext 20,60,"xx" | plottext(20,60,"xx") |

Remember - when using any graphics keywords, you must put USE GRAPHICS, or USE TURTLE at the beginning of your program, and inside any closed PROC that uses graphics keywords. At first, you should SETEXEC+ so you can tell the PROCS from the new keywords.

Q. I can't make the JOYSTICK or PADDLE commands work. Eh? (from Canada)
A. There are a couple typos in the booklet Cartridge Graphics and Sound. On pages 7 and 54, change the references to the keywords "JOYSTICK" and "PADDLE" to read "JOYSTICKS" and "PADDLES". The 'S' was left off. Also, on page 59: LINKFONT does not work from a running program. Do it in the immediate mode, after you LOADFONT("0:font.name").

Q. Page 13 of the booklet says I can speed up my MSD disk drive with the SETRECORDDELAY keyword. It doesn't work.
A. The keyword works with COMAL RANDOM (relative) files only. COMAL is too fast for the 1541.

Q. Where can I get more information about COMAL?
A. Every COMAL owner should subscribe to COMAL Today. $15. Without a subscription, you will be lost.

Q. How do I FORMAT (NEW) a disk in COMAL?
A. Using the disk drive is much easier in COMAL. That's one reason to leave the cart plugged in at all times. Page 216 of the Handbook tells all about the PASS keyword. PASS opens a command channel to the drive, passes over the command you want and then closes the channel for you. COMAL means you'll never have to OPEN1,8,15,"n0:name,id" again. With the cart, you can do this: PASS "n0:name,id"

And, finally a Mr. Richard Fader, from Fort Lee, New Jersey writes:

Q. I LISTed a cart program to the screen. Where I expected to find 'cursor graphic' symbols inside strings, I found wierd stuff like ""157""18"".

Dear Mr. Fader,
I once saw that wierd stuff, too, but then I read page 12 of the booklet. You know, where it says QUOTE'MODE(FALSE)? Now, I know what it is, and you don't. Did you ever notice that when you buy a dozen eggs, one is always broken, and in a few days it turns a real sickly green?

Thanks for writing,
Rosanna Rosannadena, COMALite.

## COMMODORE C128 COMPUTER

QUESTION: Since I currently have a C64, I am considering the possibility of upgrading to the C128. I need to know if the COMAL 2.0 Cartridge will be compatible. W.B. Colorado.

ANSWER: The cartridge has been tested with a preliminary C128 and appears to work just fine. When the C128 is released for sale we will be able to test it more thoroughly, but expect no problems. Information from Commodore indicates that while in the C64 MODE, the C128 cannot access the extra memory, 80 column screen, or faster speed of the disk drive. However, we are confident that a package can be developed for the COMAL Cartridge that will trick the computer into sharing these resources.

# COMAL CARTRIDGE PROGRAMMING TIPS

by Len Lindsay

The COMAL Cartridge is powerful. There are so many features built into it that even a 500 page book might not have room to cover everything. In this column I will explain a few features you might enjoy using.

You can put the cursor anywhere on the screen with the CURSOR command. The lines are numbered 1 - 25 with the top line as line 1. The columns are from 1 thru 40. To place the cursor at position 9 of line 5 you would use the following:

```
CURSOR 5,9
```

So far so good. But there is more. What if you want to move the cursor to line 8 but stay in the same column (position)? And you don't know what position that is? COMAL can handle it. If you specify a row or position of 0 that means keep the current value. So the way to move to line 8 without changing position would be:

```
CURSOR 8,0
```

The cursor positioning is also expanded to the INPUT and PRINT statements via the AT keyword. This is very useful. INPUT AT is nice when used with a screen layout for data input. For example:

```
INPUT AT 10,1:"Name: ": name$
```

This prints the prompt "Name: " starting at row 10 column 1 and then waits for a reply to be assigned to the variable NAME$. Also, the COMAL Cartridge provides a protected INPUT FIELD for every INPUT statement. If you have used professional programs, you probably have experienced protected INPUT fields. You will notice them right away. Keys that don't make any sense as a reply to an INPUT request are simply ignored - or even redefined. For example, CURSOR DOWN has no meaning to an INPUT request, so it is ignored. So is CURSOR UP, REVERSE ON and REVERSE OFF. In BASIC and most other programming languages it takes lines and lines of code to set up a protected INPUT field. In COMAL you get it automatically.

While waiting for a reply to an INPUT request, COMAL ignores irrelevant keys (like CURSOR UP) and redefines HOME to mean go back to the first position in the INPUT FIELD. CLEAR SCREEN is redefined to mean clear only the current INPUT field, and then go back to the first position in that field. This is very nice. You can now have impressive programs with no extra work whatsoever.

Also, keep in mind that COMAL will set up a protected INPUT field that goes from the current cursor position to the end of that line --- unless you specify otherwise. This is important. If you use this default INPUT field length, you cannot continue a reply from the end of one line to the next line. To do that you MUST specify the field length! And how do you do that you may ask.

Easy! You do it with an INPUT AT statement -- and you just saw one just above. But, in addition to specifying the line and position after the AT, you can also specify the field length. Now this can be used when you know how many characters to expect, or when you want to automatically limit the reply. For example:

```
INPUT AT 5,1,7:"Phone: ":phone
INPUT AT 20,1,10:"Name: ":name$
```

The first example can be used to prompt for a local phone number (which is always 7 digits) with the prompt "Phone: " starting at line 5 position 1. The second example requests the users name, and limits it automatically to 10 characters. If the user trys to type past the limit, all extra keys are overprinted on the final position in the field.

One more special note: if you specify an INPUT field length of 0, COMAL will only accept the RETURN key as a reply (perfect for that HIT RETURN TO CONTINUE place in your program).

Another nice thing in the cartridge - TRACE.

Any time your program is stopped: by an error, by the stop key, by a STOP statement - just type in the word TRACE, hit RETURN, and COMAL will tell you how you got to the point the program stopped

at. Perfect to answer the question "How did it get there?"

The FIND and CHANGE commands are also really nice. They make COMAL feel just like a good word processor. Find all occurances of any string of characters --- they can be variable names, keywords, or just text in a string or remark. And CHANGE allows you to not only find them, but instantly change them into some other set of characters, if you wish. Each change is optional and COMAL asks you if you wish to change the displayed line - you hit RETURN for yes and N for NO.

Finally, a few small notes for new COMAL Cartridge users:

SELECT "LP:" will now select the printer in LOWER CASE mode by default. In version 0.14 it was UPPER/GRAPHICS mode.

To clear the screen just use the PAGE command.

PI is built in now too! It is equal to a close approximation of the value of PI.

The OOOPS! key is CONTROL A. If you are editing a program line and really mess it up, simply hit CONTROL A and COMAL will restore back to its original state.

The Erase To End Of Line key is CONTROL K. This is nice when you are making a real long line shorter.

New default colors are easy to set. Set new colors with CONTROL X and CONTROL Y (followed by the CONTROL or COMMODORE key for the color). Then just type CONTROL Z. It 'burns' in the new default colors. Now whenever you use RUN/STOP RESTORE it will pop back to those new colors instead of the pale blue..

And, I hope everyone has tried the F7 function key. It is a real RUN key, since it issuesthe COMAL command: RUN. However, it can do more than just RUN the program in memory. If you look at your disks directory (with the DIR or CAT command) - you can LOAD and RUN any program by simply moving the cursor up to the line with the file name - then press F7.

The Shift RUN key has also been redefined to LOAD and RUN the first program from drive 0 on Unit 8. So, just put a menu program first on your disk, and start up with Shift RUN.

And finally - here is how to get back to the cartridge from BASIC without turning the computer off then back on:

```
SYS 50000
```

(too bad BASIC doesn't have a COMAL command).

More next time. If you subscribe to PLAYNET, you can rejoice (if you don't subscribe, now is the time to do it). PLAYNET has agreed to let THURSDAY be COMAL night. Just sign on and go to the public room named COMAL. Captain COMAL (Captain C), Geoffrey T, and me, Len L will be in and out to answer COMAL questions and spread the latest COMAL news.

## MAGIC

by Ray Carter

The purpose of this article is two-fold. First I present the conversion of an interesting little program which I originally obtained in PASCAL. The program will draw some interesting designs on the screen until it is told to stop. The second purpose is to present a conversion of the good old machine language screen dump which will do nice things on what I believe is probably the most popular non-COMMODORE printer - interface combination for the C64 (GEMINI-10X and CARDCO card?+G). The program as furnished will load the dump program for that printer and interface (of course, you can easily modify the main program to load your own dump program).The machine language screen dump takes about 2.5 minutes and generates an image which is a square representation of a full screen (i.e. if a figure fills the video screen, it will be very close to square on the printout). This is accomplished by turning each pixel of the screen into a 2 dot by 2 dot square on the paper in 960 dot per inch graphic mode. If anyone would like further information or a copy of the source assembly language they may write me at 1965 Thomas Drive, Las Cruces NM 88001. This program is on TODAY DISK #6.

# MOVING UP TO 2.0

by Len Lindsay

You switched from BASIC to COMAL 0.14 awhile ago, and wondered how you ever put up with BASIC. Now, you just moved up to the COMAL 2.0 Cartridge. You now have about twice as many commands to play with. Plus virtually everything from COMAL 0.14 is also included in the Cartridge. However, there are a few changes you should be aware of. I will point out some of these right now.

*** Graphics & Sprites Now Packages ***

Before you can give Graphics or Sprites commands you must issue a USE command:

```
USE GRAPHICS
USE TURTLE
USE SPRITES
```

*** Graphic & Sprite Commands Add () ***

Since graphics and sprites are now packages, all the 'commands' are really procedures and functions. Thus all parameters must now have () parentheses around them. For example:

| COMAL 0.14 | COMAL 2.0 |
|---|---|
| FORWARD 50 | FORWARD(50) |
| LEFT 90 | LEFT(90) |
| PENCOLOR 1 | PENCOLOR(1) |
| HIDESPRITE 2 | HIDESPRITE(2) |

*** Some Name Changes ***

In an attempt to be more compatible with LOGO and other COMAL systems, a few Graphics commands were changed:

| COMAL 0.14 | COMAL 2.0 |
|---|---|
| SETGRAPHIC 0 | GRAPHICSCREEN(0) |
| SETTEXT | TEXTSCREEN |
| FRAME | WINDOW |
| CLEAR | CLEARSCREEN |

*** Some Other Changes ***

The HOME command will return the Turtle to position 0,0, wherever that may be. After USE GRAPHICS, it is in the lower left corner - not the center of the screen. After USE TURTLE, it is in the center of the screen again, but that point is then 0,0. The left side of the screen is negative x coordinates, and the bottom is negative y coordinates.

However, COMAL 2.0 allows you to redefine, or override, any word built into a package. Thus, we can redefine HOME to move the turtle to the center of the screen while point 0,0 remains at the bottom left corner. One way to do this is by having this procedure in your program:

```
PROC home
  moveto(160,100)
ENDPROC home
```

Now, that was simple enough, right?

Turtle commands are part of both the GRAPHICS and the TURTLE packages, but the TURTLE package now also contains the LOGO style 2 letter abbreviations (ie, PU for PENUP, FD for forward, etc.).

*** WRAP MODE In TURTLE Package ***

The TURTLE package is now more like LOGO than before, complete with WRAP and NOWRAP modes. What is a WRAP you may ask? A WRAP mode means that when the turtle moves off the edge of the screen (or window if set) that it doesn't disappear - it just reappears on the other side. In COMAL 0.14 only the NOWRAP mode was implemented. Now, in COMAL 2.0 you have the choice, but the default in the TURTLE package is WRAP, not NOWRAP. The default in the GRAPHICS package is still NOWRAP.

*** SHOWSPRITE Change ***

In COMAL 0.14, everytime you identify a sprite, it was made visible automatically. This was handy, but provided problems if you wished to set up all your sprites at the start of the program, but didn't want to show them till later - or if you did a HIDESPRITE, there was no built in SHOWSPRITE to put the sprite back on. So, now in COMAL 2.0 the IDENTIFY command only sets up the image for you - it does not turn on the sprite. You must use the SHOWSPRITE command to turn it on.

Thus every COMAL 0.14 program with sprites will need a SHOWSPRITE command added after each IDENTIFY command (do this the easy way, use the COMAL 2.0 FIND command to find all the IDENTIFY commands in the program).

*** PRINTER Change ***

In COMAL 0.14 you selected the printer with this command:

```
SELECT "LP:"
```

It sent all following output to the printer. However, it opened the printer in its default mode: UPPER CASE/ GRAPHICS. Most people wished it would default to lower case mode. So, now in COMAL 2.0 you still select the printer the same way, but now it defaults to lower case mode.

*** SPLIT SCREEN MODE Change ***

Splitscreen mode in COMAL 2.0 now has 4 text lines at the top, while only 2 were shown in COMAL 0.14. Plus, in COMAL 2.0 you now can have INPUT prompts and replys appear in the split screen window, while in COMAL 0.14 you could not.

*** ENTER COMMAND Change ***

Since COMAL 2.0 adds a MERGE command to merge program segments together, the ENTER command was changed so that it automatically issues a NEW before entering the new program. This is how it should be - if you want the program to stay in memory and just add on another segment, don't use ENTER - use MERGE.

*** PENCOLOR / TEXTCOLOR Change ***

In COMAL 0.14 the PENCOLOR command was used to set not only the pencolor, but also to set the cursor color on the text screen. Now in COMAL 2.0, the cursor color on the text screen is considered separate from the pencolor on the graphics screen. This is nice - but some COMAL 0.14 programs that use PENCOLOR to change the color of the cursor on the textscreen will need to be changed. In COMAL 2.0 the TEXTCOLOR command is part of both the TURTLE and GRAPHICS package.

Also, consistent with that change, is the separation of the background and border colors for the text screen and graphics screen. BACKGROUND and BORDER now only apply to the graphics screen. To set the colors on the text screen use TEXTBACKGROUND and TEXTBORDER, also part of both TURTLE and GRAPHICS packages.

COMAL 2.0 also has one command as part of the SYSTEM package that will set all three colors: TEXTCOLORS. Here is how it sets the colors:

```
TEXTCOLORS(0,15,1)
           !  ! !
           !  ! +--cursor color
           !  +----background color
           +-------border color
```

*** QUOTE MODE Change ***

It seems that many people didn't like 'quote mode'. Actually, most HATED it. Well, COMAL 2.0 now starts off with quote mode disabled. You still can have it if you want it, but you must turn it on yourself like this:

```
USE SYSTEM
QUOTE'MODE(TRUE)
```

*** SHIFTED RETURN KEY Change ***

One habit most Commodore Computer Users have is using SHIFT RETURN. In COMAL 0.14 (and BASIC) it is different than the regular RETURN key. For instance, while editing a program line, if you decided to start over or to just leave it the way it was after changing a few things, you just hit SHIFTed RETURN and got to the next line WITHOUT the changes taking effect. NO MORE !!! Now, in COMAL 2.0 the RETURN key is a RETURN key shifted or not. That is how it should be, but old habits are already set, so watch out for it. Now, in COMAL 2.0, hit the STOP key to get to the next line.

*** AUTO MODE Changes ***

There were several changes in the AUTO line numbering mode. To stop AUTO mode in COMAL 0.14 you hit the RETURN key twice. In COMAL 2.0 you use the STOP key - more logical. When you issued the AUTO command in COMAL 0.14, it started with line 10. In COMAL 2.0 it first checks if any program lines exist yet, and if so starts with the line 10 past the last existing line - much better. And if AUTO mode prompts you with a line number that already exists - it warns you about this by printing that line number in reverse field. Finally, COMAL 2.0 lets you move up and fix a previous line and then come back to where you left off while in AUTO mode, and the line numbers are mixed up.

*** LIST TO DISK Change ***

When you LISTed a program to disk in COMAL 0.14, none of the lines were indented (to save disk space). COMAL 2.0 however, LISTs a program with indentations, no matter where the output is going - screen, printer, or disk.

One more IMPORTANT thing. Remember that COMAL 2.0 utilizes both UPPER and lower case letters to present very readable program listings. That means that a LIST to disk will result in keywords in UPPER CASE letters and variables in lower case. If you plan to ENTER that program into COMAL 0.14 remember that COMAL 0.14 does not recognize keywords in UPPER case - only in lower case (unshifted). So, when transfering a program from COMAL 2.0 into COMAL 0.14 you MUST first issue the following commands:

```
USE SYSTEM
KEYWORDS'IN'UPPERCASE(FALSE)
```

This sets the keywords into lower case. Now your program LISTings are readable to COMAL 0.14.

*** DISPLAY ***

If you are preparing an article for COMAL TODAY or other magazine about your new COMAL program, you can easily include the program listing inside the article itself without retyping it. Just issue the command:

```
DISPLAY "PROGRAM NAME"
```

It makes a sequential ASCII file of your program, complete with indentation and NO LINE NUMBERS - perfect for publications, for line numbers are usually not included in published COMAL program listings.

Of course you can also DISPLAY your program without line numbers directly to the screen or printer as well.

*** INPUT Protected ***

In COMAL 0.14 during an INPUT request, you could move the cursor up or down to any line on the screen, and when you hit RETURN you got whatever was on that line as the answer to the INPUT request. COMAL 2.0 will not let you do that. INPUT requests automatically set up a protected window for you to reply in. CURSOR UP and CURSOR DOWN are ignored. HOME puts the cursor back to the first position in the input field, and CLEAR SCREEN clears only the input field - not the whole screen. And, the input field automatically stops at the end of the screen line - you cannot continue your reply on the next line UNLESS a longer length was specified in the INPUT statement. To do this you must use a full INPUT AT statement like this:

```
INPUT AT 9,1,60: "Enter Text:": text$
         ! ! !     !             !
row num--+ ! !     !             !
col num----+ !     !             !
max length---+     !             !
prompt-------------+             !
reply assigned to this variable--+
```

The above example allows a reply of 60 characters - more than one line.

*** DIM For Strings NOT Required ***

It is no longer necessary to DIM each string variable you use. If you don't do it, COMAL will do it for you, automatically doing a DIM with a maximum length of 40. This is very handy, but remember that you must do the DIM yourself if you expect to have more than 40 characters assigned to the string.

*** FOR Loop Variable Now Local ***

COMAL 2.0 considers the control variable in a FOR loop a LOCAL variable. When the loop ends, the program no longer knows about that loop or its control variable. That is very nice, and you probably won't notice the change. But someday you will (Colin Thompson spent days over the COMAL 2.0 QUEENS program trying to make it work in COMAL 0.14 - the problem was in the FOR loop variable). Here is an example to illustrate this change:

```
X:=5
FOR X:=1 TO 9 DO NULL
PRINT X
```

COMAL 2.0 will print 5, while COMAL 0.14 will print 9. If you only utilize the FOR loop variable while inside the FOR loop, this change will not affect you.

*** IMPORT Needed In Closed PROCs ***

CLOSED procedures are very handy - they keep conflicts with independent modules to a bare minimum. In COMAL 0.14, you did not have the IMPORT capability, so they allowed all CLOSED procedures access to all other procedures in the program. Now, COMAL 2.0 makes the CLOSED procedure a bit more closed - it now doesn't know about any procedure in the outside program. To use a procedure from outside a CLOSED procedure you now must include the outside procedure name in an IMPORT statement. All this applies equally to functions.

This carries over into packages as well, since all commands in the packages are really functions and procedures. To use parts of packages from the outside program you should just issue another USE command for each package needed inside the CLOSED procedure. Or, if you wish, you could just IMPORT the package commands you need.

*** Passing An Array As A Parameter ***

If you use arrays in your programs, you may wish to pass an entire array into a procedure. In COMAL 0.14 you would use a REF parameter to do this like this:

```
DIM PLAYER$(1:3) OF 9, TABLE(1:9,1:3)
...
SHOW'IT(PLAYER$,TABLE)
...
PROC SHOW'IT(REF PLAYER$,REF TABLE)
```

In COMAL 2.0 you have the choice of using REF or not, plus you now MUST include the () parentheses section after the array name - and if it is a multiple dimension array, also include the number of commas used in its DIM statement:

```
DIM player$(1:3) OF 9, table(1:9,1:3)
...
show'it(player$(),table(,))
...
PROC show'it(REF player$(),table(,))
```

*** Printer Interface Interference ***

The Interface you use to connect a non-Commodore printer to your C64 can interfere with COMAL 2.0 (this is not a problem with Commodores printers). The most common problem is sending ESCAPE CODES to the printer. We have tested this with a BUSCARD using its parallel printer port - straight through, and could send ESC sequences to our Centronics printer fine. However, with some interfaces, they would not work (reason???). Here is how to send an ESCape 3:

```
SELECT "LP:"
PRINT CHR$(27)+"3",
```

Here is how to select the printer and send it an ESCape 3 at the same time:

```
SELECT "LP:"+CHR$(27)+"3"
  or
SELECT "LP:"27"3"
```

See note below to explain the "27".

*** CONTROL Codes ***

In COMAL 0.14 you had Quote Mode on. Thus you could put a CLEAR SCREEN inside quotes in a PRINT statement. You also had the option of using PRINT CHR$(147), since 147 is the ASCII value of CLEAR SCREEN.

In COMAL 2.0 Quote Mode defaults OFF. If you enter a program from COMAL 0.14 that uses control codes (like for Clear Screen) into COMAL 2.0, they will list differently. For example:

```
PRINT ""147"",
```

COMAL 2.0 prints a quote mark the ASCII code number and another quote mark for the control codes. To see them the old way, just turn Quote Mode back on.

*** UNIT 9 Specifications Changed ***

To pass a command to disk drive unit 9 just add comma 9 to the end:

```
PASS"I0",9
```

Most other references to unit 9 and up are different. COMAL 2.0 sees all the disk drives as numbered from 0 to 15. Drive 0 on Unit 9 is considered to be Drive 2, and drive "1:" on Unit 9 is drive "2:". This is what you use in OPEN, CAT, DIR, and other commands.

# COMAL 2.0 FROM 0.14

A COMPARISON

The chart below shows all the COMAL 2.0 keywords in the left column. The right column then indicates how that word compares to COMAL 0.14. If the word is the same in both versions the right column will say Same. If the keyword has added capabilities in COMAL 2.0 the right column will say Improved. If the keyword was not available in COMAL 0.14 the right column will say Added. If the use of the keyword is different in COMAL 2.0 than COMAL 0.14 the right column will say Different. Only one rarely used keyword (UNIT) is different in COMAL 2.0. Since COMAL 2.0 includes all the COMAL 0.14 keywords (some improved) plus more, we call COMAL 2.0 upward compatible with COMAL 0.14.

| COMAL WORD | 2.0 FROM 0.14 |
|---|---|
| ABS | Same |
| AND | Same |
| AND THEN | Added |
| APPEND | Same |
| AT | Added |
| ATN | Same |
| AUTO | Improved |
| BASIC | Same |
| BITAND | Added |
| BITOR | Added |
| BITXOR | Added |
| CASE | Same |
| CAT | Improved |
| CHAIN | Same |
| CHANGE | Added |
| CHR$ | Same |
| CLOSE | Same |
| CLOSED | Same |
| CON | Same |
| COPY | Added |
| COS | Same |
| CREATE | Added |
| CURSOR | Added |
| DATA | Same |
| DEL | Improved |
| DELETE | Improved |
| DIM | Same |
| DIR | Added |
| DISCARD | Added |
| DISPLAY | Added |
| DIV | Same |
| EDIT | Improved |
| ELIF | Same |
| ELSE | Same |
| END | Improved |
| ENDCASE | Same |
| ENDFOR | Same |
| ENDFUNC | Same |
| ENDIF | Same |
| ENDLOOP | Added |
| ENDPROC | Same |
| ENDTRAP | Added |
| ENDWHILE | Same |
| ENTER | Improved |
| EOD | Same |
| EOF | Same |
| ERR | Added |
| ERRFILE | Added |
| ERRTEXT$ | Added |
| ESC | Same |
| EXEC | Same |
| EXIT | Added |
| EXP | Same |
| EXTERNAL | Added |
| FALSE | Same |
| FILE | Same |
| FIND | Added |
| FOR | Improved |
| FUNC | Improved |
| GET$ | Added |
| GOTO | Same |
| HANDLER | Added |
| IF | Same |
| IMPORT | Added |
| IN | Same |
| INPUT | Improved |
| INT | Same |
| INTERRUPT | Added |
| KEY$ | Same |
| LABEL | Same |
| LEN | Same |
| LET | Same |
| LINK | Added |
| LIST | Improved |
| LOAD | Same |
| LOG | Same |
| LOOP | Added |
| MAIN | Added |
| MERGE | Added |
| MOD | Same |
| MOUNT | Added |
| NEW | Same |
| NOT | Same |
| NULL | Same |
| OF | Same |
| OPEN | Improved |
| OR | Same |
| OR ELSE | Added |
| ORD | Same |
| OTHERWISE | Same |
| PAGE | Added |
| PASS | Improved |
| PEEK | Same |
| PI | Added |
| POKE | Same |
| PROC | Same |
| RANDOM | Added |
| RONDOMIZE | Added |
| READ | Improved |
| REF | Same |
| RENAME | Added |
| RENUM | Improved |
| REPEAT | Same |
| REPORT | Added |
| RESTORE | Improved |
| RETURN | Improved |
| RND | Same |
| RUN | Improved |
| SAVE | Improved |
| SCAN | Added |
| SELECT INPUT | Added |
| SELECT OUTPUT | Improved |
| SGN | Same |
| SIN | Same |
| SIZE | Improved |
| SPC$ | Added |
| SQR | Same |
| STATUS | Same |
| STEP | Same |
| STOP | Same |
| STR$ | Added |
| SYS | Same |
| TAB | Same |
| TAN | Same |
| THEN | Same |
| TIME | Added |
| TO | Same |
| TRACE | Added |
| TRAP | Improved |
| TRUE | Same |
| UNIT$ | Different |
| UNTIL | Same |
| USE | Added |
| USING | Same |
| VAL | Added |
| VERIFY | Added |
| WHEN | Same |
| WHILE | Same |
| WRITE | Improved |
| ZONE | Same |

# COMPARE COMMANDS: 2.0 & 0.14

by David Stidolph

Many people who have bought the new 2.0 COMAL cartridge have previously used COMAL version 0.14 from disk. This article is to show some of the similarities, improvements, and new features of the COMAL cartridge. Since the graphics, sprites, and other machine dependent features have been put into 'packages' we will not deal with them here - only the main definition of COMAL will be talked about.

Disk loaded COMAL version 0.14 is a 'subset' of COMAL-80 (the definition of COMAL) while the 2.0 COMAL cartridge contains the full set of commands. COMAL-80 does not have any graphics or other machine dependent features in its definition, so a program written in COMAL 0.14, without graphics, should be able to be run under 2.0 COMAL.

Consider the following program. This program will take a list of names and scores, print them, and print the total and average of the scores. This program will work with either version, and do exactly the same thing, regardless of the COMAL version you are using.

```
zone 20
dim c$ of 20
total:=0
num'of'scores:=0
print "students","score"
print "========","====="
//
while not eod do
 read c$,score
 print c$,score
 total:+score
 num'of'scores:+1
endwhile
//
print "","-----"
print "total",total
print "average",total/num'of'scores
//
end
//
data "sally",67
data "sam",42
data "tom",92
data "ralph",36
data "cindy",69
```

A list of 2.0 COMAL commands are printed in this issue, with the notations "Same","Improved", and "Added". These descriptions refer to each 2.0 command compared to disk loaded COMAL. As you can see, most of the functions and many of the commands are the same in both versions. This means that many of the programs you have written in disk based COMAL can be converted to cartridge based COMAL without change.

Now that you know that many of the commands you have been using are the same, we are going to concentrate on the differences. Many of these differences do not show up when you are typing in programs (or ENTERing them), but cause problems during execution (when you RUN it).

FOR...ENDFOR

While this loop looks like its counterpart in 0.14, COMAL 2.0 has the added feature of making the control variable LOCAL to the loop. This means the loop variable used can be the same name as some other variable outside the loop without conflict. For those of use who like to use 'x' and 'y' a lot, this is a blessing. To understand the difference, look at the following program:

```
COMAL (either version)
======================
i=10
print "before loop, i=",i
for i=1 to 3 do
  print "i=",i
endfor i
print "after loop, i=",i
end
```

This program can be entered in any version of COMAL, but different outputs result depending on the version of COMAL you are using. Here is the output for both versions of COMAL on the C64

```
COMAL 0.14          COMAL 2.0
==========          =========
before loop, i=10   before loop, i=10
i=1                 i=1
i=2                 i=2
i=3                 i=3
after loop, i=4     after loop, i=10
```

PROCedures and FUNCtions

In COMAL 2.0, when you specify a procedure or function to be CLOSED, COMAL starts up a fresh name table for its use, and all variables, package commands, procedures and functions outside of the CLOSED procedure are now unknown. In order to access outside information, you now need to use the IMPORT command to put the names in the new name table. The following example shows the difference between the COMAL versions:

COMAL 0.14
==========

```
a:=1; b:=2
max(a,b)
print a
end
//
proc max(ref first,ref second) closed
 if first < second then
  swap(first,second)
 endif
endproc max
//
proc swap(ref a, ref b) closed
 t:=a; a:=b; b:=t
endproc swap
//
```

COMAL 2.0
=========

```
a:=1; b:=2
max(a,b)
PRINT a
END
//
PROC max(ref first,ref second) CLOSED
 IMPORT swap
 IF first < second THEN
  swap(first,second)
 ENDIF
ENDPROC max
//
PROC swap(ref a, ref b) CLOSED
 t:=a; a:=b; b:=t
ENDPROC swap
//
```

Both of these programs result in the same output, but notice that the COMAL 2.0 program has the keywords capitalized (you don't have to type them in capitalized, but they will be listed that way) and the added 'IMPORT' statement in the PROCedure 'max'.

ENTER

The ENTER command in 2.0 does what it does in 0.14, but first it clears any program from memory. If you want to add a listed file to the program in memory, you can use the MERGE command. Now line numbers are no longer significant.

FILE ACCESS

COMAL 2.0 has simplified and expanded I/O access (disk drive, printer, and serial port). All the new details can be found in Chapter N of The COMAL Handbook (second edition). The most important difference is that now you can access any drive (0 or 1) in any disk drive unit (8,9,10, etc.). To get a catalog of unit 9, drive 0 (a second 1541 for example) you would use the command: CAT "2:*"

```
UNIT#     DRIVE 0  DRIVE 1
=====     =======  =======
  8    =     0        1
  9    =     2        3
 10    =     4        5
 11    =     6        7
 12    =     8        9
(etc.)
```

ERROR TRAPPING

Whenever a user is asked for information, or the disk drive is accessed, the potential for error creeps in (a letter is typed in the middle of a number, or the disk drive has trouble reading a disk, etc.). These are not the only times an error can occur, but they are a prime source. The TRAP- HANDLER- ENDTRAP structure is covered in The COMAL Handbook, second edition, pages 347-353. In brief, the statements between TRAP and HANDLER are executed, and if an error occurs, execution jumps to the statements between HANDLER and ENDTRAP. Look at the following example:

```
DIM filename$ of 20
INPUT "Name of file:": filename$
TRAP
  OPEN FILE 28,filename$,READ
  CLOSE FILE 28
  PRINT "The file exists."
HANDLER
  CLOSE FILE 28
  PRINT "The file does not exist."
ENDTRAP
PRINT "All done."
```

When this codes is executed, the computer will ask you for a file name and enter the TRAPPED section of code. Now if an error occurs (like the file does not exist) execution will pass to the statements in the HANDLER section. The first statement OPENs a file. If the file exists (no error) then execution will continue, CLOSE the file, PRINT that it does exist, and bypass the code in the HANDLER section. If the file does not exist, or another error occurs (like there is no disk in the disk drive), then execution will transfer to the CLOSE statement in the HANDLER section and PRINT that the file does not exist.

PROTECTED INPUTS

Version 2.0 has a "protected" INPUT field. It does not allow cursor up or down or reverse on or off. In addition, it will not allow you to go into "quote" mode or insert mode. Finally, HOME CURSOR has been redefined during input to mean "go to beginning of the input field." CLEAR SCREEN has been redefined to mean "clear the input typed in thus far and then go to the beginning of the input field."

You can specify how long the input field is, or leave it at the default (to the end of line). This is an important feature, because it allows you to set up screens and not have someone destroy it with the CLEAR/HOME key.

TRACE

TRACE is a direct command (cannot be part of a program) that can be used after an error occurs that stops your program. It will tell you were in the program when it stopped, and, if it stopped within a procedure or function, how it got there. This is particularly useful when the program stops within a procedure that was called from another procedure. You can even type: TRACE "LP:" <RETURN> and the print out will go to the printer.

## COMAL CARTRIDGE CREDIBILITY

David Skinner sent us this note: I am having a credibility problem at times. The time it takes for the cartridge to do certain jobs is so fast that no one believes it.

## BEST SELLING COMAL BOOKS

Each issue we will report on the top 5 selling COMAL books.

There are three **NEW** COMAL books - aiming for the charts. COMAL WORKBOOK is a nice fill in the blank type workbook - a perfect match to the TUTORIAL DISK. COMMODORE 64 GRAPHICS WITH COMAL is the companion reference book to COMAL HANDBOOK. And all serious COMAL Cartridge users have been waiting for the latest release - COMAL 2.0 PACKAGES. It is a wonderful HOW TO MAKE YOUR OWN PACKAGES book written for people who already know Machine Language and wish to combine it with their COMAL programs.

DECEMBER 1984

#1 - COMAL HANDBOOK
by Len Lindsay
#2 - FOUNDATIONS IN COMPUTER STUDIES
by John Kelly
#3 - COMAL FROM A TO Z
by Borge Christensen
#4 - CARTRIDGE GRAPHICS AND SOUND
by Captain COMAL's Friends
#5a- LIBRARY OF FUNCTIONS & PROCEDURES
by Kevin Quiggle
#5b- CAPTAIN COMAL GETS ORGANIZED
by Len Lindsay

JANUARY 1985

#1 - COMAL HANDBOOK
by Len Lindsay
#2 - COMAL FROM A TO Z
by Borge Christensen
#3 - CARTRIDGE GRAPHICS AND SOUND
by Captain COMAL's Friends
#4 - STRUCTURED PROGRAMMING WITH COMAL
by Roy Atherton
#5 - FOUNDATIONS IN COMPUTER STUDIES
by John Kelly

FEBRUARY 1985

#1 - COMAL HANDBOOK
by Len Lindsay
#2 - COMAL FROM A TO Z
by Borge Christensen
#3 - CARTRIDGE GRAPHICS AND SOUND
by Captain COMAL's Friends
#4 - BEGINNING COMAL
by Borge Christensen
#5 - STRUCTURED PROGRAMMING WITH COMAL
by Roy Atherton

# COMAL KEYPAD

by Jim Borden

COMAL KEYPAD is a utility program which allows a numeric keypad to be set up below the number keys "7", "8" and "9". The numeric keypad is toggled on and off using the British Pound key. The keypad is ideal for entering numeric DATA statements, binary sprite values or inputing long lists of numbers.

After owning an orginal (ie 1978) 8K PET for four years, I was very happy to move up to a new Commodore 64. I loved the keyboard until I started entering a lot of numbers. Entering numeric data with a keypad is much faster than using the top row of keys, especially if commas and decimal points are also needed.

I decided to use the British Pound key to toggle between the normal keyboard and the numeric keypad because it is seldom (never?) used. If you must use the British Pound key, the toggle key used can be changed by POKEing a new value into the location shown on the start-up screen for your version. Do this after you load the COMAL KEYPAD program into memory. The character can be any printable character. However, keep in mind that the character will be intercepted in the IRQ routine and will never be printed.

If the RUN/STOP and RESTORE keys are pressed at the same time in 2.0 the keypad will have to be started again. This can be done by using the SYS shown on the start-up screen. Using these keys in 0.14 will not effect the COMAL KEYPAD.

After I wrote the first version of KEYPAD64 (BASIC), I was quite happy entering numeric data. However, while debugging a program, I typed "LISTxxx" (in BASIC) and got "35STxxx" because I hadn't switched the keypad off. To remind me the keypad was on I added a routine to make a click when the keypad is on. When the keypad is turned off, everything goes back to normal. With this change, this type of error does not happen often. (Only if the sound is off.) Only the changed keys will cause a click - 7, 8 and 9 will not.

Using the COMAL KEYPAD

Once the program is stored on disk, load and run the program. Now press the British Pound key and you will hear a click (if your sound is on). Now hit the "M" key. You should hear a click and a zero will appear on the screen where the cursor was. Now press the British Pound key again. Another click is heard. The British Pound key toggles the keypad on and off. I put stick-on numbers on the keypad keys. These should be available at photography or artist supply stores. If you can't find stick-on numbers, you can print your own in compressed type. I also covered mine with clear glue to prevent the numbers from rubbing off (I used "Elmer's"-no relation to me though).

After you use the COMAL KEYPAD program for a while, I don't think you'll want to enter lists of numbers without it.

```
// delete "0:comal'keypad'2.0"
// save   "0:comal'keypad'2.0"
//   by james e borden  12-9-84
//
IF PEEK($cf43)<>$68 OR PEEK($cf44)<>$a8
THEN // wrap line
  PRINT "This version ONLY for 2.0!"
  STOP
ENDIF
DIM q$ OF 1
total:=0; q$:=CHR$(34)
FOR x:=828 TO 930 DO
  READ d
  total:+d
  POKE x,d
ENDFOR x
IF total<>11201 THEN
  PRINT "error in data statements.."
  PRINT
  STOP
ENDIF
SYS $0395
PAGE
PRINT ""18"", //reverse video
PRINT "    NUMERIC KEYPAD";
PRINT " for  COMAL 2.0      "
PRINT "Use the following POKEs";
PRINT "to change data:"
PRINT
PRINT "POKE 913,ORD(""@"")   ";
PRINT "To change toggle"
PRINT SPC$(17),"character where '@' is"
PRINT SPC$(17),"the new chracter."
PRINT
```

```
PRINT "POKE 905,x      ";
PRINT "Where x=0 for normal"
PRINT SPC$(17),"keyboard and x=1 for"
PRINT SPC$(17),"numeric keypad. (For"
PRINT SPC$(17),"programed control.)"
PRINT
PRINT "Hit [RUN/STOP][RESTORE]";
PRINT "to kill KEYPAD."
PRINT
PRINT "'",CHR$(PEEK(913)),"' is toggle";
PRINT "key now. ",
PRINT "Hit that key and"
PRINT "try the u, i, o, j, k,";
PRINT "l and m keys."
//ML data below
DATA 164,198,240,67,185,118,2,162
DATA 7,221,138,3,240,5,202,16
DATA 248,48,52,224,7,208,12,198
DATA 198,173,137,3,73,1,141,137
DATA 3,16,11,173,137,3,240,31
DATA 138,9,48,153,118,2,169,15
DATA 141,24,212,169,65,141,18,212
DATA 160,32,136,208,253,169,64,141
DATA 18,212,169,0,141,24,212,104
DATA 168,104,170,104,64,0,77,74
DATA 75,76,85,73,79,92,76,60
DATA 3,120,162,2,189,146,3,157
DATA 67,207,202,16,247,88,96
```

VERSION FOR COMAL 0.14:

```
// delete "0:comal'keypad.14"
//  by james e. borden  1984
// save   "0:comal'keypad.14"
dim q$ of 1
total:=0; q$:=chr$(34)
for x:=51968 to 52097 do
 read d
 total:+d
 poke x,d
endfor x
if total<>16573 then
 print "Error in data statements..."
 stop
endif
sys 52064
//
print "          COMAL KEYPAD 0.14"
for x:=1 to 4 do print//blank lines
print "Use the following pokes to chang
e data:" // wrap line
print
print "poke 52051,ord(",q$,"@",q$,")
To change toggle" // wrap line
print "         character where '@' is"
print "         the new character."
print
print "poke 52043,x"
print "         x=0 for normal keyboard"
print "         x=1 for numeric keypad."
print
print "'",chr$(peek(52051)),"' is toggl
e key now. Hit that key and try",//wrap
print "the u, i, o, j, k, l and m keys."
print chr$(13)+"Keypad is active now..."
// ml data below
data 32,135,234,164,198,240,67,185
data 118,2,162,7,221,76,203,240
data 5,202,16,248,48,52,224,7
data 208,12,198,198,173,75,203,73
data 1,141,75,203,16,11,173,75
data 203,240,31,138,9,48,153,118
data 2,169,15,141,24,212,169,65
data 141,18,212,160,32,136,208,253
data 169,64,141,18,212,169,0,141
data 24,212,96,0,77,74,75,76
data 85,73,79,92,169,51,141,171
data 235,169,48,141,165,235,32,202
data 120,162,86,189,49,234,157,144
data 203,202,16,247,169,0,141,219
data 203,169,203,141,220,203,169,144
data 141,229,16,169,203,141,230,16
data 88,96
```

## ANOTHER FUN LETTER

We get lots of nice mail. Many are like the following letter from R.W. in Utah.

Where have you been all my life? About two months ago I was exposed to COMAL for the first time, having never even heard of COMAL before then. It is incredible! It is what BASIC should have been all along. And I'm just talking about the 0.14 version! Now I am writing programs exclusively in COMAL. I even dream in the language. It can do things I wouldn't begin to know how to do in Commodore BASIC even using machine language subroutines. But I want more. I want to get the 2.0 version with all the added features for $128.90. That's cheap at twice the price (but wait until I get one before raising it). However, could you please answer a few questions:

* How much faster is the 2.0 version over 0.14? About twice as fast.

* Are programs written in COMAL 0.14 compatible with the 2.0 version and vice-versa as long as the instructions are implemented in 0.14? Yes, the COMAL is compatible. The add on GRAPHICS and SPRITES environment (package) had some minor changes, most notable that now all graphics and sprites keywords with parameters require parentheses () around them.

# CHEER UP - IT COULD BE WORSE

or

**TALES OF THE WEST COAST COMMODORE SHOW**

by Colin Thompson

[ED NOTE: You will have to read this several times to catch all the fun Colin has included. This is Colin at his finest. Please read it in that light]

It sounded too good.

The man on the phone had a distinct English accent and was using it well. "Malcolm Lowe," he said, "and I'd like to invite you to speak about COMAL at the West Coast Commodore Convention in February".

Following the standard offer of airfare and hotel expenses, he asked if I could make it. "Well, let's see," I mumbled, as I thumbed through my appointment book. Ah, here's February, and - ooops, I'm due to speak in Calgary on that weekend. Calgary!? What lunatic agreed to go to Calgary in the winter?

"This lunatic," I said. "What's this about a lunatic?" Malcolm politely asked. "Sorry, just reading out loud, Malcolm. I seem to be scheduled for Canada then, but let me get back to you in a couple days. Maybe something will change."

"Something will definately change," I swore under my breath. Bye, Malcolm. Quick, what's the number of the Calgary group?

Two months and several phone calls later, I meet Len Lindsay at the San Francisco airport. I've decided that this was a good opportunity to preach COMAL to my favorite kind of group, a captive audience, so I've invited Len to lend some authority to the event. This is Len's first visit to San Fransisco, so I play tour guide during the Limo ride to the hotel. He's amazed at the sight of a strange city, jammed with concrete ribbons of freeways. I'm amazed at the weather- it's not raining.

Following the ritual of tips, check-in, credit cards, tips, bellmen, baggage, tips, long corridors and tips, we set out in search of some friends, who, we have on good authority, are staying in the same hotel. Our first target is Jim Butterfield, who hasn't yet checked in. Next is the Computer Curmudgeon, Mindy Skelton, who has already taken up residence, but cannot be found. We eat.

While enjoying our lunch of red snapper that looks and tastes just like sole, we remember that Randy Chase was going to attend. The front desk (the computer, really) informs us it has no record of a Mr. R. Chase, of Oregon. Strike three.

Idle curiosity moves us to the mezzanine, where the show is being set up. Chaos. Wires, lights, boxes, frantic people. We get our badges (yes, we don't need any stinkin' badges, but we pin them on anyway). Now armed with a visual "right to exist", we enter the center of the melee, the combat zone, the main floor. Any trade show, the day before it opens, looks just like this. Kinda like a zoo, populated by a commitee. We walk.

Steve of PP+S tells me he has a great new product. I never get to see it. Tom Lynch, President of the Valley Group invites me to his booth. He has written a COMAL Cartridge program that is running on his 1702, shouting the advantages of belonging to THE user group of north LA. I liked the program.

We walk in search of the PlayNET booth. Len assures me he can recognize Mindy - he met her last year. Cathy "Miss PlayNET", says Mindy hasn't been there for a while, so we turn to walk farther. Wham! Bang! I'm surrounded by arms. Much hugging. I nearly fall down. Mindy helps me to my feet. Len, ever the gentleman, makes a tardy and now unnecessary introduction. "Let's eat. I want to tell you both about PlayNET," commands the Curmudgeon. We head for the coffee shop, following in the wake of a person who has enough excess energy to sell to Three Mile Island. Seated in the now familiar setting of the coffee shop, Mindy tells us that Randy can't make it. Something about having a magazine to get out on time. She begins an account of her new existance as a PlayNET Junkie as Jim B walks in. Len excuses himself to confer with The Man.

Mindy continues, undaunted. I listen. PlayNET, it seems, (I have not seen it at this point), was created by God, Herself,

and given to the World Of Commodore, (which She also created). Mindy prefers PlayNET to sex or food. I begin to wonder about Mindy. She says we MUST go to the PlayNET booth after the phone lines are installed and she will give us both an electronic tour of it's chambers of delight. "I can't wait," I respond.

Len returns, and we repair to our respective rooms to "dress" for the evening's entertainment. Commodore is holding a reception for user group functionaries, followed by a gala given by the WCCA organizers for the vendors and speakers. I surely wouldn't want to miss either, so I suit up in my Sunday finest, replete with tie. We three meet in the lobby. They don't seem to recognize me. I introduce myself. They are clad in the latest version of "California Casual"- jeans, tee shirt and running shoes. I'm wearing a tie. I feel out of place, they let me come along anyway.

I know we are close to our target destination, a suite on the 7th floor, when I see Jim B holding forth in the crowded corridor. We squeeze by and enter the room. It's packed with user group luminaries and Commodore officials. The officials shout about their newfound "support" of user groups. The luminaries ask detailed questions. The officials mumble into their drinks.

I look at the sign-in register and discover, to my delight, that some folks that I correspond with are here in person. I start looking at nametags. Most of the nametag owners stare back. The next hour is spent in delightful conversation with John Zacharias of Sacramento, Craig Borden of Tulsa, Jane Campbell of San Diego, Shelly Roberts of Nu Yawk, and Rich Tsukiji of Oregon. Shelly introduces me to Wes James, a commercial programmer and NYCIG's Guru. Wes asks what's a COMAL. Wrong question, Wes. I talk, he listens.

Wes finally dismisses me with an "I prefer compiled BASIC" and we adjourn to the ballroom, downstairs. The Commodore Contingent is led by Pete Baczor and Jim Gracely, earnestly explaining why they think the new model 128 will become the sweetheart of the marketplace. They haven't heard of an "Amiga". They DO know all about the new lap computer, the LCD. Whenever asked a technical question about the LCD, they wrinkle their brows, think a minute, then point, in tandem, to a young man near the punch bowl. "He knows. Ask him. He designed it," they whisper. Pete plays Oliver Hardy and Jim does Stan Laurel. I enjoyed the act.

I don't much like parties. I tend to find a corner and just hang out, wishing I owned a good Lap Computer to pass the time. The corner I picked was keeping company with Guy Wright, Mr. RUN Magazine. He likes corners also. We talked for nearly an hour, while observing the high priesthood of The Church of Commodore give blessings on their adoring flock.

All good things must end (Book of the Kernal 3:12), and thus it came to be; I was "rescued" from my corner and made to join 11 other revellers on a Mission From God - it's time to eat. Dick Immers knows a great Tibetian Barbeque that's "just around the corner". We set out on foot, following him. He leads us on a tour of San Francisco's alternate lifestyle district. Len stares. I cringe. We all hurry, but this doesn't seem to be Little Tibet. We pass the Hard Rock Cafe. The line goes around the corner, so we settle on a tacky little Italian place that features a 99 year old man playing the theme from the Godfather on a 99 year old piano. We are the only patrons. Maybe the rest of the city knows something about this place that we don't.

Dick tells how much fun he had writing his book on the 1541. Wes announces he has completed the first true CAD system for the 64 called CAD GEM. It's being marketed by Commodore and will be out in a few weeks. It's a real 'wire frame', three axis, hidden line design system with a screen size of 2 megabytes - all for only $90. I write a check on the spot and forgive him for his comment about compiled BASIC.

Next stop on our whirlwind tour is the bar in the hotel. There our party fragments and I wobble to my room, hoping to catch a few hours sleep to prepare myself for the rigors of opening day.

What rigors? I have nothing to do. Our lecture is on the following day, Sunday,

so I rise with the roosters and wobble to the coffee shop. Len has arranged to share booth space with Loadstar and I decide to sit in on some of the morning presentations.

I've attended dozens of trade shows, but have never actually listened to a lecture. Amazing. Well, my initiation began at 10 AM when Laurel and Hardy gave their presentation. They have a new program designed to support Commodore User Groups. Apparently Commodore just found out about User Groups. The main thrust of their new program is an eight page newsletter - called Input/Output. The issue I got seemed to be slanted toward Output. Many of the pages were given over to the important task of revealing the software available for those dynamite computers, the Plus Four and the Commodore 16. Great stuff, and Commodore will actually send this newsletter, free of charge to any user group that applies for "qualification" and gains "approval". I fill out the application form with hands trembling in anticipation.

Stan tells us about Commodore's long term plans. (In the long term, we all die). Olly tells us about the new C128. It has three "configurations", not to be confused with "modes". Semantics are important to someone in Upper "Management". We are introduced to the first configuration - called the "Sixty-four mode-er-configuration". It is completely compatable with all existing C64 software and hardware. Next, the 128 mode-etc. This is a real gem, we hear. It has a lot of memory - 122K free, and uses the 1571 Disk Drive. The 1571 is a Real Neat Thing. It can tell what kind of a diskette is in the drive and alter it's bus transfer rate to match the microprocessor needed to RUN the program.

Did you get that?

Maybe I should explain the third mode-configuration. It's called CP/M and Commodore is really pushing it. They have convinced Thorne-EMI to rewrite the Perfect series CP/M programs to run on the 128 in CP/M mode. CP/M is what's called 'disk intensive' - it uses the disk drive all the time, so the bus transfer rate had better be fast or it will take all day to write a memo on WordStar. We all know that Commodore has the slowest disk drives in this universe and probably the next. In order to use CP/M, the 1541 had to be coaxed into HyperDrive. That's where the 1571 comes in. It has a three speed automatic transmission.

In the C64 mode it behaves just like a 1541 - 300 Characters Per Second transfer rate. When the 128 mode is selected, the drive steps up to 1500 CPS. That's a step in the right direction, about 8 years too late. Finally, the CP/M mode needs some REAL speed and gets it - 3500 CPS. This may prove to be a workable combination. Time will tell.

Speaking of time, I wonder how many hours will be spent trying to jimmy the Operating System into letting the C64 mode use the 3500 CPS disk rate? Carl Sagen's "Billions and Billions" sounds like a low estimate.

The LCD looks like a good machine. It has a big, fold up LCD screen display. It does not have Easyscript built into ROM. Too bad. I would have bought the cute lil' bugger if I could write ES articles on the plane, or some other inconvienent place. They showed off a 3 1/2 inch Sony drive that plugged into the LCD. That drive intrigued me. I wanted one or two for my C64. If they actually release it, I might be able to use it on the 64. At least that's what the young man who designed the system told us. You could plug in a 1541 also, but not a monitor or TV set. The LCD's built in software looked similar to the NEC's built in software. After hearing Commodore talk about all the good things the LCD will do, I think I may be able to write text files on it, then download them to my 1541 at home. Then I could bring the files in EasyScript. That's almost a workable plan. Nice going, Commodore.

The last 15 minutes of the presentation were spent answering questions from the audience. Johnny Carson never had to contend with an audience this hostile. It seemed every other question was "What about the Amiga?". And of course, every other answer was "I don't know anything about an Amiga, or a Lorraine, or any other 68000 based microprocessor that the trade press has prematurely announced". End of story? Not hardly.

All throughout the presentation, Commodore's young, unnamed genius, tried to make the C128 do something - anything. It, of course, would not budge. Murphy strikes. When the show concluded, I wandered up the the stage to get my hands on a keyboard. I remember the dismay I felt when I first touched a Plus Four. I'm sorry to report that Commodore gave the keyboard contract to the same firm that ruined the Plus Four. Too bad. I'll just wait for the phantom Amiga. Jim B swears it will be out by next February, or April at the latest.

I don't recall much of what went on the rest of that day. Just a blur of people without names, another attempt at finding Little Tibet, and then Bed.

Bright And Early Sunday Morning. Len and I are scheduled to speak at 11 AM, an hour after the show opens it's doors. Once again, I'm sporting my tie, etc. Over breakfast we plan how to divide our talk. I'll open with a five minute overview, then hand it over to Len. He does 15 minutes on the State OF COMAL, then I do 20 minutes of technical discussion. That leaves Len 5 minutes to describe the function of his COMAL Users Group, USA. With our plans firmly set in mind, we walk to the conference room.

Eleven AM. The computer is set up and all the hardware is working. On the overhead projector, the COMAL Cartridge is showing off. The room will hold 400. 40 of the chairs are filled and we begin. I start to speak. The room reverberates with sound of empty chairs. I continue, regardless. Thirty minutes later, I conclude my five minute intro, and hand it over to Len. He talks until Malcolm politely unplugs the mike. Our time is up. The crowd surges to the stage and we begin to answer all the questions that were considered 'too stupid' to ask in front of everyone. We politely answer them until Malcolm's security agents escort us from the hall. Alas, 45 minutes is simply not enough time.

I approach the President of the WCCA and politely inform him I would consider doing this again, only if I'm given four hours, on a Saturday afternoon to make my presentation. He agrees. Len and I pack up and fly to Santa Monica. Now he sees Real freeways. He is amazed. As the cab nears my apartment, I realize that we have not seen a drop of precipitation in three days. The temperature is 72 degrees, and this is February. I wonder what it's like today in Calgary.

## REVIEW - GRAPHIC PRIMER

Book / disk set - (special now $14.95)
reviewed by David Stidolph

Three different textbooks have been published to help people learn COMAL. However, none of these textbooks teach anything about COMAL's graphics or sprites. Now, thanks to Mindy Skelton, COMALites can learn about the graphics abilities of their computer. The book, over 80 pages in length, is a good introduction to drawing graphics and sprite control.

Beginning with a short introduction to programming and procedures, it moves quickly to "turtle" graphics (turtle graphics is the system of drawing that COMAL uses). The approach in teaching graphics is usually done while the student actually tries the commands as they read about them, and this book follows that format. Each command is described in the order that you should learn them, so there is little need to "jump" around the book.

Sprite control is one of the hardest graphics features to learn, but Mindy's book comes to the rescue with step-by-step instructions on what a sprite is and how to create and move it. Not one or two, but three ways of creating sprites is shown.

Many sample programs, and useful procedures and functions are presented, each with explanations on what they do. Finally, the book has a good index and a complete glossary of ALL graphics and sprite commands, including the GETCOLOR command, which was not known to be a valid function until this book was published.

Since the book comes with a matching disk that contains all the programs from the book typed in, very little typing is needed to try the examples. The disk has a menu of the sample programs, so you just LOAD and RUN that program, and you can select any of the examples to RUN.

GRAPHIC PRIMER is a good tutorial for COMAL 0.14 graphics. I highly recomend it.

# COMAL 2.0 CARTRIDGE MEMORY MAP



| Label | Address | Comments |
|---|---|---|
| d6510 | 0000 | 6510 on-chip data-direction register |
| r6510 | 0001 | 6510 on-chip 6-bit i/o/ map-register: |
| prproc | 0002-0004 | chain of local names (prepass) |
| integr | 0005 | fp work |
| page | 0006 | current memory map |
| pagept | 0007-0008 | pointer used by load/store/exec |
| pagex | 0009 | overlay for load/store/ exec routines |
| pagey | 000a | overlay used for control of jump table |
| p6510 | 000b | old c64-overlay for control of jump table |
| resol | 000c | graphics resolution |
| gcolh | 000d | graphics pencolor*16 |
| loclpt | 000e-000f | chain of old variable descriptions |
| forpt | 0010-0011 | stack entry chain |
| sctype | 0012 | type of symbol from scanner |
| tansgn | 0013 | tan sign / comparison evaluation flag |
| code | 0014 | used to hold a generated code |
| cpnt | 0015 | pointer to code buffer, cdbuf |
| sprog | 0016-0017 | pointer to start of program |
| svars | 0018-0019 | pointer to start of variable table |
| sstack | 001a-001b | pointer to start of stack |
| smax | 001c-001d | pointer to top of memory |
| exinf | 001e | inf for result expression from expr |
| lnlen | 001f | length of line to be executed |
| npnt | 0020 | pointer to name |
| tpnt | 0021 | pointer to string |
| index1 | 0022-0023 | utility pointer |
| index2 | 0024-0025 | utility pointer |
| resm1 | 0026 | product area for multiplication |
| resm2 | 0027 | |
| resm3 | 0028 | |
| resm4 | 0029 | |
| resm5 | 002a | |
| datapt | 002b-002c | current data pointer |
| stos | 002d-002e | pointer to top of stack |
| sfree | 002f-0030 | pointer to free area of var.res |
| prgpnt | 0031-0032 | pointer to start of line |
| codpnt | 0033 | pointer to code during execution |
| sclsd1 | 0034-0035 | old sfree (closed) |
| sclsd2 | 0036-0037 | old stos (closed) |
| inf1 | 0038 | used for operand |
| inf2 | 0039 | checking |
| inf3 | 003a | |
| q1 | 003b-003c | short span work areas |
| q2 | 003d-003e | |
| q3 | 003f-0040 | |
| q4 | 0041-0042 | |
| q5 | 0043-0044 | |
| copy1 | 0045-0046 | work for copy: from |
| copy2 | 0047-0048 | to |
| copy3 | 0049-004a | length |
| bus | 004b | bus=0: bus idle<br>bus<>0: bus active |
| stinf | 004c | information for statement: |
| excinf | 004d | execution information |
| tempf3 | 004e-0053 | miscellaneous floating point work area |
| escape | 0054 | stop key flag |
| | 0055 | not used |
| oldov | 0056 | old overflow (rounding) |
| tempf1 | 0057-005b | miscellaneous f.p. work area (5 bytes) |
| tempf2 | 005c-0060 | miscellaneous f.p. work area (5 bytes) |
| ac1e | 0061 | accumulator 1 exponent |
| ac1m1 | 0062 | mantissa 1 |
| ac1m2 | 0063 | mantissa 2 |
| ac1m3 | 0064 | mantissa 3 |
| ac1m4 | 0065 | mantissa 4 |
| ac1s | 0066 | sign |
| degree | 0067 | series evaluation constant pointer |
| bits | 0068 | accum#1: hi-order (overflow) |
| ac2e | 0069 | accumulator 2 exponent |
| ac2m1 | 006a | mantissa 1 |

| | | |
|---|---|---|
| ac2m2 | 006b | mantissa 2 |
| ac2m3 | 006c | mantissa 3 |
| ac2m4 | 006d | mantissa 4 |
| ac2s | 006e | sign |
| arisgn | 006f | sign comparison, acc#1 vs acc#2 |
| facov | 0070 | accum#1: lo-order (rounding) |
| polypt | 0071-0072 | pointer to polynom. |
| asave | 0073 | save for .a (call/goto) |
| xsave | 0074 | save for .x (call/goto) |
| psave | 0075 | save for .p (call/goto) |
| indpnt | 0076 | pointer to last code where an address was loaded |
| scflag | 0077 | flags in scanner: |
| lnno | 0078-0079 | line number |
| movead | 007a-007b | address for move |
| txtlo | 007c | address of text for prtxt |
| txthi | 007d | |
| xx | 007e-007f | current x (graphics) |
| yy | 0080-0081 | current y (graphics) |
| grwk1 | 0082-0083 | |
| grwk2 | 0084-0085 | |
| grwk3 | 0086-0087 | |
| excflg | 0088 | flags: |
| charpt | 0089 | pnt to inbuf |
| char | 008a | char from inchar |
| rndx | 008b-008f | random number seed |
| status | 0090 | i/o operation status |
| stkey | 0091 | stop key flag |
| svxt | 0092 | temporary |
| verck | 0093 | load or verify flag |
| c3p0 | 0094 | ieee buffered char flag |
| bsour | 0095 | char buffer for ieee |
| syno | 0096 | cassette sync # |
| xsav | 0097 | temp for basin |
| ldtnd | 0098 | how many files open |
| dfltn | 0099 | default input device # |
| dflto | 009a | default output device # |
| prty | 009b | cassette parity |
| dpsw | 009c | cassette dipole switch |
| msgflg | 009d | os message flag |
| ptr1 | 009e | cassette error pass 1 |
| ptr2 | 009f | cassette error pass 2 |
| time | 00a0-00a2 | 24 hour clock in 1/60 seconds |
| pcntr | 00a3 | cassette stuff |
| firt | 00a4 | |
| cntdn | 00a5 | cassette sync countdown |
| bufpt | 00a6 | tape buffer pointer |
| inbit | 00a7 | rs232 rcvr input bit storage |
| bitci | 00a8 | rs232 rcvr bit count in |
| rinone | 00a9 | rs232 rcvr flag for start bit check |
| ridata | 00aa | rs232 rcvr byte buffer |
| riprty | 00ab | rs232 rcvr parity storage |
| sal | 00ac | pointer: tape buffer/ screen scrolling |
| sah | 00ad | |
| eal | 00ae | |
| eah | 00af | |
| cmp0 | 00b0 | tape timing constant |
| temp | 00b1 | tape timing constant |
| tape | 00b2 | start of tape buffer |
| bitts | 00b4 | rs232 trns bit count |
| nxtbit | 00b5 | rs232 trns next bit to be sent |
| rodata | 00b6 | rs232 trns byte buffer |
| fnlen | 00b7 | length of current file name |
| la | 00b8 | current file logical address |
| sa | 00b9 | current file secondary address |
| fa | 00ba | current file primary address |
| filadr | 00bb-00bc | current file name address |
| roprty | 00bd | rs232 trns parity buffer |
| fsblk | 00be | cassette read block count |
| mych | 00bf | serial word buffer |
| cas1 | 00c0 | cassette manual/ controlled switch |
| stal | 00c1 | tape start address |
| stah | 00c2 | |
| memuss | 00c3-00c4 | tape load temps |
| lstx | 00c5 | key scan index |
| ndx | 00c6 | key buffer pointer |
| rvs | 00c7 | reverse field on flag |
| indx | 00c8 | byte pointer to end of line for input |
| lsxp | 00c9 | start of screen input (row) |
| lstp | 00ca | start of screen input (column) |
| sfdx | 00cb | shift mode on print |
| blnsw | 00cc | cursor blink enable |
| blnct | 00cd | counter to flip cursor |
| gdbln | 00ce | old char before blink |
| blnon | 00cf | on/off blink flag |
| crsw | 00d0 | input/get flag |
| pnt | 00d1-00d2 | pointer to start of line where cursor is flashing. |

| | | |
|---|---|---|
| pntr | 00d3 | column position where cursor is flashing. |
| qtsw | 00d4 | flag for quote mode |
| lnmx | 00d5 | current screen line length (39/79) |
| tblx | 00d6 | line number where cursor is flashing |
| data | 00d7 | temp data area |
| insrt | 00d8 | number of insert keys pushed to go |
| wrptb | 00d9-00f2 | line flags+endspace |
| user | 00f3-00f4 | screen editor color pointer |
| keytab | 00f5-00f6 | keyboard decode table |
| ribuf | 00f7-00f8 | rs232 input buffer address |
| robuf | 00f9-00fa | rs232 output buffer address |
| frekzp | 00fb-00ff | free kernal z-page space |
| stack | 0100-01ff | system stack |
| fbuffr | 0100-010f | fpasc work area (15 bytes) |
| ertlen | 0200 | length of ertext |
| ertext | 0201-0249 | buffer to hold error message |
| conpnt | 0250 | old prgpnt |
| conflg | 0251 | old excinf |
| concod | 0252 | old codpnt |
| confor | 0253-0254 | old forpt |
| fpwork | 0256 | |
| extrom | 0257 | external rom flag (0=no, '1'=yes) |
| ieeein | 0257 | ieee installed (0=no, '1'=yes) |
| lat | 0258-0262 | table of la's |
| fat | 0263-026c | table of fa's |
| sat | 026d-0276 | table of sa's |
| keybuf | 0277-0280 | keyboard buffer queue (fifo) |
| memstr | 0281-0282 | start of memory |
| memsiz | 0283-0284 | top of memory |
| timout | 0285 | ieee time out defeat |
| color | 0286 | active color nybble |
| gdcol | 0287 | original color under cursor |
| hibase | 0288 | bas location of screen |
| kbflim | 0289 | size of keyboard buffer |
| rptflg | 028a | key repeat flag |
| rptcnt | 028b | repeat speed counter |
| delay | 028c | repeat delay counter |
| shflag | 028d | keyboard shift key/ctrl key/c= key |
| lstshf | 028e | last keyboard shift pattern |
| keylog | 028f-0290 | vector: keyboard table setup |
| shmode | 0291 | 0=shift disabled $80=shift enabled |
| autodn | 0292 | auto scroll down, 0=on |
| m51ctr | 0293 | 6551 control register image |
| m51cdr | 0294 | 6551 command register image |
| m51ajb | 0295-0296 | non-standard bps (time/2-100) usa |
| rsstat | 0297 | 6551 status register |
| bitnum | 0298 | number of bits left to send |
| baudof | 0299-029a | baud rate: full bit time (microsec) |
| ridbe | 029b | index to end of input buffer |
| ridbs | 029c | start of input buffer (page) |
| rodbs | 029d | start of output buffer (page) |
| rodbe | 029e | index to end of output buffer |
| irqtmp | 029f-02a0 | holds irq-vector during tape i/o |
| enabl | 02a1 | rs232 enables |
| caston | 02a2 | tod sense during cassette i/o |
| kika26 | 02a3 | temp storage for cassette read |
| stupid | 02a4 | temp d1irq indicator for cassette read |
| lintmp | 02a5 | temp for line index |
| palnts | 02a6 | flag: 0=ntsc, 1=pal |
| filnam | 02a7-02dd | used for storage of file name or disk command |
| rangno | 02de | line # range pointer |
| rangpt | 02df | line # range pointer |
| ranges | 02e0-02ff | line # ranges |
| ierror | 0300-0301 | vector: print basic error message |
| | 0302-0306 | not used |
| num2 | 0307-030b | fp work area (print using) |
| sareg | 030c-0313 | not used |
| cinv | 0314-0315 | irq ram vector |
| cbinv | 0316-0317 | brk instr ram vector |
| nmivct | 0318-0319 | nmi ram vector |

| | | |
|---|---|---|
| iopen | 031a-031b | open routine vector |
| iclose | 031c-031d | close routine vector |
| ichkin | 031e-031f | chkin routine vector |
| ickout | 0320-0321 | ckout routine vector |
| iclrch | 0322-0323 | clrchn routine vector |
| ibasin | 0324-0325 | chrin routine vector |
| ibsout | 0326-0327 | chrout routine vector |
| istop | 0328-0329 | stop routine vector |
| igetin | 032a-032b | getin routine vector |
| iclall | 032c-032d | clall routine vector |
| usrcmd | 032e-032f | for machine language monitor |
| iload | 0330-0331 | load routine vector |
| isave | 0332-0333 | save routine vector |
| | 0334-033b | unused |
| tbuffr | 033c-03fb | tape i/o buffer |
| | 03fc-03ff | unused |
| screen | 0400-07e7 | text screen memory area |
| | 07e8-07f7 | unused |
| sprpnt | 07f8-07ff | sprite data pointers (unused) |
| mbegin | 0800-0803 | start of memory |
| mbegn1 | 0804 | start of name table |
| mbegn2 | 0805 | start of stacks |

| | | |
|---|---|---|
| rsibuf | c000-c0ff | rs232 input buffer |
| rsobuf | c100-c1ff | rs232 output buffer |
| stdpck | c200-c5e7 | variables for standard packages |
| inbuf | c5e8-c660 | input buffer (121 characters) |
| cdbuf | c661-c75f | code buffer |
| txt | c760-c7af | string constant buffer |
| flevel | c7b0 | for/trap nesting level during prepass |
| q6 | c7b1-c7b2 | temporary |
| q7 | c7b3-c7b4 | temporary |
| q8 | c7b5-c7b6 | temporary |
| q9 | c7b7-c7b8 | temporary |
| | c7b9-c7bb | unused |
| spsav | c7bc | save of .s during execution |
| scinf | c7bd-c7be | name pointer |
| auto1 | c7bf-c7c0 | current line number for auto |
| autost | c7c1-c7c2 | step for auto |
| dstart | c7c3-c7c4 | start of data queue |
| tabset | c7c5 | value of last zone statement |
| altpos | c7c6 | position in select output file |
| intrno | c7c7-c7c8 | procedure given by interrupt statement |
| errpnt | c7c9 | char pos of error |
| norint | c7ca-c7cb | normal interrupt vector |
| safe | c7cc | safe status |
| mainrv | c7cd | main revision |
| subrv | c7ce | sub revision |
| testrv | c7cf | test version |
| msglin | c7d0-c7d1 | address of message line |
| upper2 | c7d2 | copy of borge |
| extprc | c7d3 | flag for loading of external proc/func |
| extcnt | c7d4-c7d5 | nesting level of external proc/func |
| ssize | c7d6 | 40 col./80 col. |
| lunit | c7d7 | last disc (status) |
| borge | c7d8 | special flags for listing |
| openfl | c7d9 | flag used by copen |
| dfunln | c7da | length of default unit text |
| dfunit | c7db-c7dc | default unit (power up value: .byte '0') |
| defout | c7dd | select output flag |
| trapvc | c7de-c7df | pageb; error handler |
| extnvc | c7e0-c7e1 | pageb; external load |
| usrqvc | c7e2-c7e3 | pageb; interrupt facility |
| iertxt | c7e4-c7e6 | error message data |
| igetln | c7e7-c7e8 | pagea; input command line |

| | | |
|---|---|---|
| isavec | c7e9-c7ea | pagec; save additional info |
| iloadc | c7eb-c7ec | pagec; load additional info |
| ifnkey | c7ed-c7ee | pagea; handle function keys |
| libpt | c7ef | ptr to place for next library description |
| liblo | c7f0-c7f9 | |
| libhi | c7fa-c803 | |
| libpag | c804-c80d | |
| modet | c80e-c817 | open mode for files |
| countt | c818-c821 | table of byte count for files |
| stt | c822-c82b | status for opened files |
| recotl | c82c-c835 | table of record pos. for files |
| recoth | c836-c83f | |
| ppage | c840 | overlay to peek/poke/sys |
| norest | c841 | <>0: disable stop/restore |
| loadin | c842 | <>0: loading comal program |
| unitfl | c843 | 0: simp.dev; 1: drive; 2: cassette |
| mode | c844 | file mode |
| cstat | c845 | status of comal program |
| lstflg | c846 | bit vector for rcreat: |
| lpmode | c847 | default printer open mode |
| lpsa | c848 | default printer secondary address |
| lpfa | c849 | default printer unit |
| recdel | c84a | record positioning delay |
| endadr | c84b-c84c | top of ram |
| headln | c84d | power on message flag |
| kwtab | c84e-c84f | keyword table (pagea) |
| dfbord | c850 | default border color |
| dfback | c851 | default background color |
| dfforg | c852 | default foreground color |
| acbord | c853 | actual text border |
| acback | c854 | actual text background |
| keylen | c855-c864 | lengths of function key definitions |
| klen | c865 | # of chars left of def. |
| kpnt | c866-c867 | pointer to key definitions |
| definp | c868 | select input flag |
| hz50 | c869 | 0=60 hz, 1=50 hz tod |
| | c86a-c87a | reserved for future use |

subroutines to use in assembler coded subroutines in comal:

| | | |
|---|---|---|
| cold | c87b | cold start of comal |
| warm | c87e | warm start of comal |
| call | c881 | jsr to another page. |
| goto | c884 | jmp to another page. |
| load | c887 | load from pagex |
| store | c88a | store to pagex |
| exec | c88d | jsr to pagex |
| ldac1 | c890 | load ac1 |
| ldac2 | c893 | load ac2 |
| fndpar | c896 | find parameter (assembler calls) |
| copy | c899 | copy area towards lower addresses |
| copydn | c8a2 | copy area towards higher addresses |
| fpadd | c8ab | load ac2 and add ac2 to ac1 |
| fpadd2 | c8ae | add ac2 to ac1 |
| fpahf | c8b7 | add 0.5 to ac1 |
| fpsub | c8c0 | load ac2 and sub ac2 from ac1 |
| fpsub2 | c8c3 | sub ac2 from ac1 |
| fpmul | c8cc | load ac2 and multiply ac2 by ac1 |
| fpmul2 | c8cf | mul ac2 by ac1 |
| fpdiv | c8d8 | load ac2 and divide ac2 by ac1 |
| fpdiv2 | c8db | div ac2 by ac1 |
| mul10 | c8e4 | multiply ac1 by 10.0 |
| div10 | c8ed | divide ac1 by 10.0 |
| stac1 | c8f6 | store ac1 |
| c1t2 | c8f9 | copy ac1 to ac2 |
| c2t1 | c902 | copy ac2 to ac1 |
| fpneg | c90b | negate ac1 |
| fpsgn | c914 | sign of ac1 |
| fpsin | c91d | sine of ac1 |
| fpcos | c926 | cosine of ac1 |
| fpsqr | c92f | square root of ac1 |
| fptan | c938 | tangent of ac1 |
| fppow | c941 | raise ac2 to the power of ac1 |
| fpatn | c94a | arctangent of ac1 |
| fpexp | c953 | raise ac1 to the power of e |
| fplog | c95c | logarithm base e of ac1 |
| fprnd | c965 | compute pseudo-random number (range 0 to 1) |
| fpcom | c96e | compare number to ac1 |
| trunc | c977 | convert ac1 into integer (-32768 .. 32767) |
| fpintg | c980 | convert ac1 into integer (-2↑24 .. 2↑24-1) |
| fpinta | c989 | convert ac1 into integer (0 .. 65535) |
| intfp | c992 | convert integer into fp in ac1 |

| | | |
|---|---|---|
| fpasc | c99b | convert ac1 into ascii equivalent (str$) |
| val | c9a4 | convert decimal string into binary in ac1 |
| popa1 | c9aa | pop ac1 |
| popa2 | c9b3 | pop ac2 |
| pusha1 | c9bc | push ac1 |
| pushrl | c9c5 | push real number |
| pshint | c9ce | float & push integer (-32768 .. 32767) |
| intfpa | c9d7 | float & push integer (0 .. 65535) |
| excgst | c9e0 | allocate local storage |
| excrem | c9e9 | reclaim local storage |
| restop | c9f2 | allocate global storage |
| runerr | c9fb | go to comal error handler |
| crdt | ca01 | read character |
| space | ca04 | write space |
| cwrt | ca06 | write character |
| cchkin | ca09 | select input file |
| cckout | ca0c | select output file |
| cclrch | ca0f | clear channel |
| cfname | ca12 | parse & copy file name |
| copen | ca19 | open file |
| cclose | ca1c | close file |
| crlf | ca1f | output cr and lf |
| getlin | ca22 | input keyboard line |
| reset | ca29 | reset program pointers |
| dummy | ca2f | empty subroutine (rts) |
| comal | ca30 | go to comal editor |
| excute | ca36 | execute code in cdbuf |
| jload | ca3d | load comal program |
| arrlen | ca44 | compute number of elements in array |

## ANATOMY OF A 2.0 CARTRIDGE

by David G. Sommers

My child's curiosity got the better of me yesterday. I had to check out the inside of my COMAL 2.0 cartridge to see how they handled 64K in one cartridge and if adding additional EPROM/ROM is possible.

First I want to say that it appears to be well designed and executed. Excellent little board and materials.

Inside are eight chips; four 27128 EPROMs, a 74LS04, a 74LS08 a 74LS138, and a 74LS161. The 161 is used as a kind of programmable latch. At hardware reset its output is cleared to select (through the 138) the first EPROM and assert the correct controls on the C=64 to accept external ROM control.

The overlaying of the four EPROMs is controlled by the 161 being reprogrammed by the code in the active EPROM. The C64 already partially decodes various I/O and memory areas for external options. This is used to advantage. Some sections of the 04 and 08 are used as a simple I/O port in the I/O 1 memory area of the C64 ($DE00-$DEFF). When a new EPROM needs to be kicked in, the code in the EPROM in use writes the correct byte to this I/O port causing the new EPROM to be engaged. The COMAL cartridge then appears only as a 16K ROM based at $8000.

I refer you to the Commodore 64 Programmers Reference Manual page 266 for a basic memory map.

The rest of the EPROM decoding is controlled through the 138 which is on the address bus as well as being controlled by the 161 programmable counter/latch.

As is, this cartridge design does not allow the addition of any more EPROM/ROM. However, expansion is possible. To accommodate additional EPROM/ROM it is probably necessary to build a larger version of the board with the added EPROM/ROM sockets wired in parallel; except for pin twenty which would go to pins eleven and ten of the 138 for overlays five and six respectively.

A bit harder perhaps, is the addition of a second 161 to accomodate the D2 signal.This latched signal will now be routed to the presently grounded pin three of the 138. Other latch schemes are possible. This is just one way.

[ED NOTE: I also have been told that the 16K EPROMs could be replaced by 32K EPROMs - giving 64K added 'expansion' area.]

## VIEWPORT AND THE SCREEN DUMP

B.H. of Illinois found an application of the VIEWPORT command. If you do a graphics screen dump (CONTROL D) while a viewport is in effect - only the area inside the viewport is dumped, the rest of the screen is ignored. If you do want the entire screen dunped, make sure to go back to full screen with: VIEWPORT(0,319,0,199).

 - C64 COMAL 0.14 MEMORY MAP

```
0       6510 on-chip data direction register
1       6510 on-chip 5-bit input/output register
        (this controls the current memory setup--be careful if you change it)
43      temporary storage of error number about to be generated
51-52   pointer to base of current string.
56-57   start of program (start value 35153)
58-59   start of variables (start value 35153)
60-61   start of name table (start value 35153)
62-63   end of name table (start value 35154)
64-65   start of variables (start value 35161)
66-67   bottom of dim variables(start value 45056)(reset by new/run/chain)
        (reset takes value from 2066-2067)
68-69   highest location used by comal (start value 45056)
        (reset by new/chain)
        (reset takes value from 2066-2067)
97      floating point accumulator#1: Exponent
98-101  floating point accumulator#1: Mantissa
102     floating accumulator#1: Sign
103     pointer: Series Evaluation Constant
104     floating point accumulator#1: Overflow Digit
105     floating point accumulator#2: Exponent
106-109 floating accumulator#2: Mantissa
110     floating point accumulator#2: Sign
111     Sign Comparison Result: Accum.#1 verses #2
112     Floating Accumulator#1: Low-Order (Rounding)
113-114 Pointer to the cassette buffer
144     Kernal I/O status word
145     reverse field (0=off 1=on)
146     timing constant for tape
147     flag: 0=Load, 1=Verify
148     flag: Serial Bus-Output Char. Buffered
149     Buffered Char. for Serial Bus
150     Cassette Sync number
151     temp data area
152     0=screen 1=printer // output location - see also 4348
153     Default input device (0)
154     Default Ouput Device (3)
155     Tape Character Parity
156     flag: Tape byte-recieved
158     tape pass 1 error log
159     tape pass 2 error log
160-162 real time jiffy Clock
165     cassette sync countdown
166     pointer: Tape I/O Buffer
167     RS-232 Input Bits / Cassette Temp
168     RS-232 Bit count / Cassette Temp
169     RS-232 Flag: Check for start bit
170     RS-232 Input byte buffer / Cassette Temp
171     RS-232 Input Parity / Cassette Short Count
173-174 pointer: tape buffer / screen scrolling
176-177 tape timing constants
178-179 pointer: start of tape buffer
180     RS-232 out bit count / Cassette Temp
181     RS-232 next bit to send / Tape EOT flag
182     RS-232 out byte buffer
183     Length of Current File Name
184     Current logical file number
185     Current secondary address
```

```
186      Current device number
187-188  pointer: current file name
189      RS-232 out parity / Cassette temp
190      cassette read/write block count
191      serial word buffer
192      tape monitor interlock
193-194  I/O start address
195-196  Tape load temps
197      last key pressed (255=none)
198      keystroke buffer count
204      0=cursor enable 1=cursor disable
205      cursor timing countdown
206      character under cursor
207      last cursor blink :: on / off
208      input from screen / from keyboard
209-210  current physical screen line address
211      position of cursor on line
212      quote mode (0=off 1=on)
213      current physical screen line length
214      line cursor is on (0-24)
215      last inkey/checksum/buffer
216      number of inserts outstanding
217-242  screen line link table / line wrap table
243-244  pointer: current screen color map start
245-246  vector: keyboard decode table
247-248  pointer: rs-232 input buffer
249-250  pointer: rs-232 output buffer
251-252  pointer: next data statement location
256-511  micro processor stack area
512-600  system input buffer
601-610  Kernal table: active logical file numbers
611-620  Kernal table: device number for each file
621-630  Kernal table: secondary address for each file
631-640  keyboard buffer
645      flag: kernal variable for IEEE timeout
646      current pencolor
647      current color under cursor (background color)
648      top of screen memory page
649      size of keyboard buffer
650      repeat enable: 128=repeat any key after approx 1/2 second
651      repeat speed counter
652      repeat delay counter
653      special keys (0=none 1=shift 2=commodore key 4=control key)
654      last keyboard shift pattern
655-656  vector: keyboard table setup
657      flag: 0=disable shift key, 128=enable shift key
658      flag: auto scroll down, 0=on
659      RS-232: 6551 control register image
660      RS-232: 6551 command register image
661-662  RS-232: non-standard BPS (time/2-100) USA
663      RS-232: 6551 status register image
664      RS-232: number of bits left to send
665-666  RS-232 baud rate: full bit time (micro seconds)
667      RS-232 index to end of input buffer
668      RS-232 start of input buffer (page)
669      RS-232 start of output buffer (page)
670      RS-232 index to end of output buffer
671-672  holds IRQ vertor during tape I/O
673      RS-232 enables
674      TOD sense during cassette I/O
675      temp storage for cassette read
```

```
676     temp D1IRQ indicator for cassette read
677     temp for line index
678     PAL/NTSC flag, 0=NTSC / 1=PAL
679-701 file name for disk access (including CAT)
702-733 Free -- We think.
734-739 Used during DEL.  Holds deletion line range (10000 added to
        line numbers.
740-774 Free -- We think.
775-787 Unknown, but not free.
788-789 IRQ vector
790-791 BRK insturction vector
792-793 NMI vector
794-795 OPEN   vector
796-797 CLOSE  vector
798-799 CHKIN  vector
780-781 CHKOUT vector
782-783 CLRCHN vector
784-785 CHRIN  vector
786-787 CHROUT vector
808-809 STOP   vector (scan for stop key pressed)
810-811 GETIN  vector
812-813 CLALL  vector
814-815 User Defined vector
816-817 LOAD   vector
818-819 SAVE   vector
820-827 UNUSED! --7 bytes--
828-1019  disk / cassette buffer
1024-2023 text screen memory
2024-2039 free memory
2040-2047 sprite pointers (not applicable normally)

2049        basic program 'sys 2063'
2066-2067   top address space available on power-up (only used once)
2024-2065   UNUSED (by COMAL) --22 bytes--
2068-2762   start of comal keyword table
            format: 1 byte length of word
            followed by command word (cbm format)
4321        linefeed after carrage return if not zero (0)
4325-4326   old IRQ vector
4348        0=screen 1=printer // output location - see also 152 ($0098)
            routine to send carrage return (and linefeed if necessary)
6608        sys to this location to cause the error number in loc 43 ($002b)
11500-11513 code to reset dim variables and high mem pointers
11605       new text IRQ
11902       new graphics NMI
11924       new graphics IRQ
11951       new text NMI
12002       number of border color used by RUN/STOP RESTORE
12007       number of background color used by RUN/STOP RESTORE
12012       number of pencolor used by RUN/STOP RESTORE

            COMAL starts here
12036-12089 setup new interrupt vectors:
            hardware irq vector ---> 11605
            nmi irq vector --> 11951
12090-12112 copy basic rom to hidden ram underneath
12113-12116 switch basic rom out
12117-12121 set background color to blue
12122-12126 set border color to light blue
12127-12159 print 'initial greeting screen'
12543       prints the '9902' portion of 9902 bytes free.
```

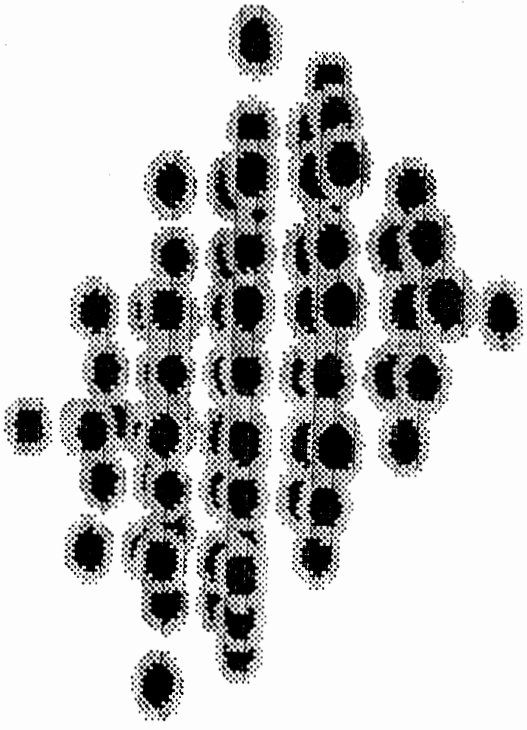
```
12547       general print message routine use to print greeting screen
            uses 117, 118 as indirect pointers to ascii bytes
            of text to print // message ends with a $00 (hex)
27255-27256 x coordinate of turtle
27258       turtle size
27260       y coordinate of turtle
27261       type of graphics screen now in use - hi-res (0) or multi-color (1)
27276       sprite on or off bits
27277-27278 heading of turtle
27295       turtle state - visible (1) or invisible (0)
27333       turtle pen state - down (1) or up (0)
29260       Base of Character ROM used for PLOTTEXT
            (208 for upper/graphics, 216 for upper/lower)
34643-35151 logon message/ tokenized display line last entered
34869       text entered in quote mode
34891       ascii (petascii) display line last entered
35153-45056 comal program work space
45056       top of programing space
45057-49151 BASIC routines copied to ram underneath (math, input, etc.)
45969       Fix to Float
47095       Float to Fix
47187       Perform (subtract)
45290       Perform (add)
47594       Perform (log)
47659       Perform (multiply)
47870       Divide by 10
47890       Perform (divide)
48034       Memory to Floating Point Accumulator #1
48124       move Floating Point Accumulator #2 to #1
48140       move Floating Point Accumulator #1 to #2
48185       Perform (sgn)
48216       Perform (abs)
48219       compare Floating Point Accumulator #1 to memory
48283       Float to Fix
48332       Perform (int)
48605       Float to ascci
49076       Perform (negative)
49133       Perform (exp)
49152-49215 sprite image 0
49216-49279 sprite image 2
49280-49343 sprite image 2
 ...   ...   continued every 64 bytes
52608-52671 sprite image 55
52672-52735 sprite image 56
52736-53247 rs-232 buffer (512 bytes)
53248       Start VIC chip
            ==> refer to Programmers Reference Guide page 321
            Start of first Character Generator ROM (UPPER/GRAPHICS)
54272       Start SID chip
            ==> refer to Programmers Reference Guide page 324
55296       start of screen character colors & graphics screen
            (hi-res color map under the I/O)
            Start of second Character Generator ROM (lower/UPPER)
56256-56319 start of turtles current image - just a guess
56320       Start CIA1 chip (keyboard CIA chip)
            ==> refer to Programmers Reference Guide page 328
56321       joystick & button - port 1
56328-56331 hardware clock / timer
56333       poke ,1 = disable timer a interrupt // poke ,129 = enable
56335       part of clock / timer
57344-65535 start of bit map for graphics screen
```

# DRAW MOLECULE

by David Stidolph

The following program can, using a definition which you store in a disk file, draw a three dimensional picture of a molecule. The program was originally written for the Macintosh computer, and then translated to 2.0 COMAL. The disk file contains the X, Y, and Z coordinates, as well as the size of each atom in the molecule. All the figures are relative, and any size system can be used, so long as you are consistent. The program "create'silicon" will create a file for a silicon lattice of the size you want.

The drawing program will, when run, ask you for the name of the molecule data file, and load the information. Once the information is in the computer, the program will ask you the angle you want to view it at, how far away you want to be from it and how much, if any, you want the molecules to be magnified. It will then rotate the molecule and draw it. Once done, the program will wait for you to press a key, and ask if you want to continue. If you press C to continue, the program will ask for a new angle, etc., so you can redraw the same molecule without reloading the data file.



**SCREEN DUMP OF SILICON MOLECULE**

```
// delete "0:create'silicon"
// save   "0:create'silicon"
// create file for DRAW MOLECULE
PRINT "Generate Nx, Ny, Nz";
PRINT "Silicon unit cells:"
PRINT "2,1,3",
INPUT AT 0,1: "": nx,ny,nz
INPUT "file name: ": name$
sz:=.15
OPEN FILE 1,name$,WRITE
FOR i:=0 TO nx DO
  FOR j:=0 TO ny DO
    FOR k:=0 TO nz DO
      WRITE FILE 1: i,j,k,sz
      IF i<>nx AND j<>ny AND k<>nz THEN
        WRITE FILE 1: i+.5,j+.5,k,sz
        WRITE FILE 1: i+.5,j,k+.5,sz
        WRITE FILE 1: i,j+.5,k+.5,sz
        WRITE FILE 1: i+.5,j+.5,k+1,sz
        WRITE FILE 1: i+.5,j+1,k+.5,sz
        WRITE FILE 1: i+1,j+.5,k+.5,sz
        WRITE FILE 1: i+.25,j+.25,k+.25
        ,sz // wrap line
        WRITE FILE 1: i+.25,j+.75,k+.75
        ,sz // wrap line
        WRITE FILE 1: i+.75,j+.25,k+.75
        ,sz // wrap line
        WRITE FILE 1: i+.75,j+.75,k+.25
        ,sz // wrap line
      ENDIF
    ENDFOR k
  ENDFOR j
ENDFOR i
CLOSE
END
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
// delete "0:draw'molecules8"
// save   "0:draw'molecules10"
// originally written for the
// Machintosh computer by
//     E. Kirkland 4-JUL-84
//printed BYTE 2/85 pages 251-259
// rewritten for COMAL 2.0 (c64)
//     By: David Stidolph
USE graphics
USE system
graphicscreen(1)
background(0)
border(0)
window(10,300,-10,205)
pencolor(11)
textcolors(5,5,11)
DIM x(0:500), y(0:500)
DIM z(0:500), s(0:500)
DIM y'(0:500), z'(0:500)
DIM s'(0:500), x'(0:500)
// ask what to do
textscreen
PAGE
```

```
PRINT "Data file name: dat.silicon",
INPUT AT 0,17,20: "": name$
// read atomic coordinates from
// data file and scale
OPEN FILE 1,name$,READ
xmin:=1e+25; xmax:=-xmin
ymin:=xmin; ymax:=xmax
zmin:=xmin; zmax:=xmax; n:=0
//
WHILE NOT EOF(1) DO
  n:+1
  READ FILE 1: x'(n),y'(n)
  READ FILE 1: z'(n),s'(n)
  IF x'(n)>xmax THEN xmax:=x'(n)
  IF x'(n)<xmin THEN xmin:=x'(n)
  IF y'(n)>ymax THEN ymax:=y'(n)
  IF y'(n)<ymin THEN ymin:=y'(n)
  IF z'(n)>zmax THEN zmax:=z'(n)
  IF z'(n)<zmin THEN zmin:=z'(n)
ENDWHILE
CLOSE
//
PRINT n;"atomic coordinates"
xmin:=.5*(xmax+xmin)
ymin:=.5*(ymax+ymin)
zmin:=.5*(zmax+zmin)
REPEAT
  PRINT
  PRINT "Please wait..."
  FOR l#:=0 TO n DO
    x(l#):=x'(l#); y(l#):=y'(l#)
    z(l#):=z'(l#); s(l#):=s'(l#)
  ENDFOR l#
  clear
  fill(160,100)
  PRINT
  PRINT "Azimith, Polar angles: 45,55",
  INPUT AT 0,24: "": phi,theta
  PRINT "Viewing distance: 7",
  INPUT AT 0,19: "": viewd
  PRINT "Magnification: 1.2",
  INPUT AT 0,16: "": smag
  phi:=phi*PI/180; theta:=theta*PI/180
  cp:=COS(phi); sp:=SIN(phi)
  ct:=COS(theta); st:=SIN(theta)
  rotate
  sort
  scale'picture
  plot'molecules
  WHILE KEY$<>""0"" DO NULL
  WHILE KEY$=""0"" DO NULL
  textscreen
  PRINT
  PRINT "Press C to continue"
  REPEAT
    c$:=KEY$
  UNTIL c$<>""0""
UNTIL NOT c$ IN "cC"
// rotate molecule around its center
PROC rotate
  PRINT "rotating..."
  FOR i#:=1 TO n DO
    xa:=x(i#)-xmin; ya:=y(i#)-ymin
    x(i#):=cp*xa+sp*ya
    y(i#):=-sp*xa+cp*ya
    ya:=y(i#); za:=z(i#)-zmin
    y(i#):=ct*ya+st*za
    z(i#):=-st*ya+ct*za
  ENDFOR i#
ENDPROC rotate
// sort by depth(shell sort)
PROC sort
  PRINT "sorting..."
  igap:=INT(n/2)
  WHILE igap>=1 DO
    FOR i#:=igap+1 TO n DO
      FOR j#:=i#-igap TO 1 STEP -igap DO
        jg:=j#+igap
        IF y(j#)<=y(jg) THEN
          j#:=1 // quit outer loop
        ELSE
          swap(x(j#),x(jg))
          swap(y(j#),y(jg))
          swap(z(j#),z(jg))
          swap(s(j#),s(jg))
        ENDIF
      ENDFOR j#
    ENDFOR i#
    igap:=INT(igap/2)
  ENDWHILE
ENDPROC sort
// for perspective projection
// and scale coordinates
PROC scale'picture
  scale:=-1e+25; smax:=scale
  FOR i#:=1 TO n DO
    ya:=1/(viewd-y(i#))
    x(i#):=x(i#)*ya
    z(i#):=z(i#)*ya
    s(i#):=s(i#)*ya
    IF scale<ABS(x(i#)) THEN
      scale:=ABS(x(i#))
    ENDIF
    IF scale<ABS(z(i#)) THEN
      scale:=ABS(z(i#))
    ENDIF
    IF smax<s(i#) THEN smax:=s(i#)
  ENDFOR i#
  scale:=100/(scale+.5*smax*smag)
ENDPROC scale'picture
//
PROC filloval(x'loc,y'loc,rad,clr)
  pencolor(-1)
  circle(x'loc,205-y'loc,rad)
  paint(x'loc,205-y'loc)
  pencolor(clr)
  fill(x'loc,205-y'loc)
ENDPROC filloval
```

```
//
PROC plot'molecules
  fullscreen
  FOR i#:=1 TO n DO
    ix:=x(i#)*scale+160
    iy:=z(i#)*scale+100
    ir:=s(i#)*scale*smag
    plot'draw
  ENDFOR i#
ENDPROC plot'molecules
//
PROC plot'draw
  filloval(ix,iy,ir,15)
  filloval(ix,iy,ir*.8,12)
  filloval(ix,iy,ir*.6,11)
ENDPROC plot'draw
//
PROC swap(REF a,REF b) CLOSED
  t:=a; a:=b; b:=t
ENDPROC swap
```

This program is on TODAY DISK #6.

## COMAL CARTRIDGE CONTROL

George Jones reports: Here are a couple of things I discovered about the cartridge. The 'control T' key combination in COMAL 2.0 is a delete character. If the cursor is at the far left margin of a line of text, 'control T' will delete all characters to the right of the cursor as long as it is held down. I discovered that by accident. A 'NEW' issued in 2.0 also discards all packages in memory at that time. It appears that the cartridge turns the C64 into an ASCII machine as many of the 'control x' functions work in BASIC as well.

I'm doing my best to get others started now and continuing my own COMAL education. I am also in the process of converting to 2.0 a rather extensive program I started in 0.14. The power that 2.0 gives me now requires some reconsideration of my original concept and the program will be even more comprehensive. It's not going to be earthshaking but still, it will be a considerable effort for someone who has been computing for only two years. I hope it will have commercial application & that's why I'm not saying anything else about the program for now. You'll be among the first to know when I'm ready to announce it.

## COMAL CARTRIDGE SHADOW LETTERS

by Bob Hoerter

This program shows how easy it is to make nice looking headlines using the TEXTSTYLE and the PLOTTEXT commands that are part of the cartridge GRAPHICS package.

By changing the size of the FOR loops as well as the X and Y increments / decrements some very different effects can be achieved.

The PROC fancy'print could easily be changed to accommodate different type sizes by adding these parameters and using them in the TEXTSTYLE statement.

```
USE graphics
graphicscreen(1)
border(0)
background(1)
//
fancy'print(60,150,"COMAL")
fancy'print(100,105,"is")
fancy'print(45,60,"number")
fancy'print(120,10,"1")
WHILE KEY$=""0"" DO NULL
END "End of program."
//
PROC fancy'print(xcor,ycor,t$) CLOSED
  IMPORT textstyle,pencolor,plottext
  textstyle(4,5,0,1)
  pencolor(0)
  x:=1; y:=1
  FOR lp#:=1 TO 4 DO
    plottext(xcor+x,ycor-y,t$)
    x:+.5; y:+1
  ENDFOR lp#
  pencolor(12)
  FOR lp#:=2 TO 3 DO
    plottext(xcor+1-lp#,ycor+1-lp#,t$)
  ENDFOR lp#
ENDPROC fancy'print
```

## SPECIAL COMAL CARTRIDGE SECTION

QUESTION: How does one open the bloody cartridge case without breaking it?

ANSWER: Opening the HANDIC Case is very hard (I know, we have to open them here) but fear not, there is no need to ever open it since there are no empty sockets or switches or anything user modifiable inside.

QUESTION: Can an owner of a cartridge make a duplicate? I have access to an EPROM burner and 32K EPROMs?

ANSWER: No. It is illegal to copy the cartridge EPROMS as they are copyrighted and no permission has been granted to allow copying.

## CARTRIDGE CASE - THE PLASTIC HOUSING

J.S. of Florida writes: I have purchased the COMAL Cartridge and wish to express a few thoughts. LAUREL: Darn good job, one of the best - programming wise - implementation of a language on the C64. DART: While the software is excellent, the HANDIC casing is terrible. It does not fit well in the cartridge port, it is friction fitted together (falls apart when you don't want it too, and stick shut when they should open). The Commodore cartridge case fits well and is held together with a screw. REPLY: The first cartridges use the HANDIC case because that is the case that they chose (we had to use it). Glad you approve of the Commodore case - that is how the next batch will come.

## CARTRIDGE LABEL

QUESTION: I just received the COMAL 2.0 Cartridge which was labelled COMAL 80 for Commodore 64. Is this the correct cartridge. I ordered the COMAL DELUXE Cartridge Package.

ANSWER: Yes, it is the correct cartridge. COMAL is short for COMAL-80, it's official name. The 80 has no special meaning other than specifying the current standard (The first COMAL standard was COMAL-74).

## POWER SUPPLY PROBLEMS

Several people have reported problems with their 2.0 cartridges. The problem seems to be that many of the Commodore 64 power supplies barley put out enough power for the computer. The 2.0 cartridge contains four EPROMS as well as some support chips, and this requires slightly more power than a simple game cartridge. This power problem can cause video problems, and even stop the computer. This is the same possible problem BUSCARD users face. A solution for this is getting a new power supply. There are a few on the market that look good. We have ordered two of each kind we saw advertised for testing. Since we haven't received them yet we can't report on them here. But if you are interested here is a source we found:

Computer Place, 23914 Crenshaw Blvd, Torrance, CA 90505 has C64 power supply with extra power for $39.95.

## PATTERN MATCHING DIRECTORIES

J.S. of OHIO provides this comment: I am beginning to suspect that COMAL has a lot more to it than what is in the 470 page COMAL HANDBOOK. I found one that is very interesting and useful. Type:

CAT "<filename>,<filename>,..."

Fill a whole line if you wish. When you press return you get the disk header, the listed filenames and the number of blocks remaining.

Thank you. What you have found is that the 'pattern matching' can be extended beyond one string. End any <filename> with a * and get all files listed that start with the same letters before the *.

## BUS EXPANDERS & COMAL CARTRIDGE

The CARDBOARD 5 bus expander from CARDCO seems to have problems with the COMAL 2.0 Cartridge. The bus expander that we recommend (and use ourselves here) is the APROSPAND-64 for $39.95. This four slot expander is available from APROPOS TECHNOLOGY - 1-800-962-5800 / 1-800-962-3800.

## CARTRIDGE IS FINAL

QUESTION: Will I need to upgrade my COMAL 2.0 Cartridge when the new ROM version comes out?

ANSWER: No. The ROM version is identical (except for the start up screen) to the cartridge you have. We did ship 10 preliminary cartridges out to software developers - but those have all been upgraded now (I hope). However, if you really do need an empty socket, the ROM version cartridge (black) has that, while the EPROM version (gray) does not. No exchange plan is available at this time. All purchasers were given the option of returning the cartridge if the empty socket was wanted.

## COMAL 2.0 DRIVE NUMBERS

QUESTION: How do I access the drives on my device 9 disk drive?

ANSWER: COMAL 2.0 refers to your disk drives as numbers 0 thru 15 and does not specify a unit number. Unit 8 drives (the default) are drives 0 and 1. Unit 9 drives are drives 2 and 3, etc. Thus to see a catalog of the disk in drive 0 of unit 9 you would use:

```
CAT "2:"
```

The exception to this is the PASS command, which requires a unit number after the command string if a drive other than 8 is to be used:

```
PASS "I0",9
```

## BATCH FILES

QUESTION: What is the command for executing BATCH files? M.L. Arizona.

ANSWER: To execute the commands in a BATCH file use the SELECT INPUT command:

```
SELECT INPUT "NAME"
```

The file should be an ASCII file created previously with PRINT FILE statements or other means. COMAL will treat all text from the file as if it were being typed at the keyboard (just like a giant keyboard buffer).

## COMAL CARTRIDGE INPUT CLARIFIED

QUESTION: The information in the COMAL HANDBOOK is incomplete and misleading. It should indicate that the default for this command is to accept input fields which go to the end of the screen line. That is, if you have a prompt length n characters, INPUT will only allow you to type in 40-n characters. Of course you can specify the input field length, but the instructions for doing so are not clear. S.L. New Jersey

ANSWER: Unfortunately, the COMAL HANDBOOK was written using a preliminary Cartridge, where the INPUT was allowed to continue on the next line. However, UNICOMAL (authors of COMAL CARTRIDGE) changed this as you have eloquently noted. To get around this limit use the INPUT AT statement and specify a length (60 in the example below):

```
INPUT AT 9,1,60: "Enter text> ": text$
```

## 170K DYNAMIC KEYBOARD

QUESTION: I would like to have my program PEEK a machine language program and prepare a DATA statement FILE to be added to the program automatically. How? L.B. North Carolina

ANSWER: COMAL will not allow you to modify a running COMAL program. However, by creating a BATCH file of DATA Statements you could MERGE them onto the program automatically with a SELECT INPUT command at the end of your program. Use PRINT FILE statements to create the SEQ BATCH FILE. Make sure each 'line' includes a different line number, then the word DATA, followed by the numbers, separated by commas. Don't exceed 79 characters per 'line'. If you name the file "DAT.ML" then use this command at the end of your program: SELECT INPUT "DAT.ML"

## COMAL CARTRIDGE IN THE SX64

QUESTION : Does the COMAL Cartridge work with the SX-64?

ANSWER: Yes. That is what we used to demonstrate the Cartridge at the World Of Commodore II show.

## CARTRIDGE SIMULTANEOUS ACTION

The COMAL Cartridge has some pretty amazing features built in - standard features. It is a giant leap ahead of what other computer publications write about. A typical example is in the March 1985 issue of COMPUTE'S GAZETTE on page 10 where they say: "There's another technique which is even closer to simultaneous action. But it requires an intermediate to advanced knowledge of Machine Language (ML)..." They then describe what we have built into COMAL with the 2.0 Cartridge! With COMAL it requires absolutely NO knowledge of Machine Language, and is easy. It is the SPRITE ANIMATION and BUILT IN INTERRUPT DRIVEN MUSIC. I don't know how much longer the power of the COMAL Cartridge can be kept secret.

## COPY RELATIVE FILES / COMAL 2.0

QUESTION: When I saw in Datamost's Book INSIDE COMMODORE DOS that a 1541 could not copy relative files I was skeptical to say the least. On an impulse I decided to get my 1541's from the shelf where they have sat since returning from repair (I had bought MSD drives in the meantime) and see for myself. On another impulse I wrote the test program in COMAL. D.T. Illinois

ANSWER: Thank you for finding that COMAL can copy a relative file. I Believe that COMAL 2.0 CAN treat a relative file as a sequential file. Maybe that is why it could do a COPY of it.

## CHARACTER ROM

QUESTION: How does one access the character ROM of the C64 from the COMAL 2.0 Cartridge? I need the data that determines the shape of alpha-numeric characters. G.J. Nebraska.

ANSWER: Try SETPAGE(2). This should flip in the character ROMs at $D000-$DFFF. Make sure you issue the command USE SYSTEM first since SETPAGE is in the SYSTEM package. You also could try GETCHARACTER in the FONT package. See details on these in the CARTRIDGE GRAPHICS AND SOUND book. Also, look at the program CREATE'FONTS on Cartridge Demo Disk #2.

## PASS COMMAND FOR DRIVE 9

The COMAL HANDBOOK was written with a preliminary release of COMAL 2.0, and thus several errors crept in. PASS will work with Disk Drive Unit 9, but using the following syntax:

```
PASS "<command string>",9
```

The 9 is the device number. For example, the following will initialize Drive 9:

```
PASS "i",9
```

## BOOKS INCLUDED IN DELUXE CART PAK

QUESTION: What books are included in the Deluxe COMAL 2.0 Cartridge Package?

ANSWER: The Deluxe Cartridge Pak includes the COMAL HANDBOOK second edition and CARTRIDGE GRAPHICS AND SOUND. It also includes two disks: Cartridge Demo Disks #1 & #2.

## COMAL 2.0 GEMINI SCREEN DUMP

Here is a graphics screen dump for the COMAL Cartridge to the Gemini printer.

```
// delete "0:proc.gemini'dump"
// list   "0:proc.gemini'dump"
PROC gemini'dump CLOSED
  z:=ZONE
  ZONE 0
  USE system
  setpage(0)
  OPEN FILE 78,"u4:/t+/s5",WRITE
  PRINT FILE 78: ""27"",CHR$(65),""8""
  bitmap:=$e000
  FOR col#:=0 TO 39 DO
    PRINT FILE 78: ""27"",CHR$(75),CHR$
    (200),""0"", // wrap line
    FOR row#:=24 TO 0 STEP -1 DO
      FOR bit#:=7 TO 0 STEP -1 DO
        b#:=PEEK(bitmap+col#*8+row#*320
        +bit#) // wrap line
        PRINT FILE 78: CHR$(b#),
      ENDFOR bit#
    ENDFOR row#
    PRINT FILE 78: ""10""
  ENDFOR col#
  PRINT FILE 78: ""27"@"
  CLOSE FILE 78
  ZONE z
ENDPROC gemini'dump
```

## CARTRIDGE AVAILABLE SEPARATELY

QUESTION: Is a manual or instructions provided wih the $99.95 plain COMAL Cartridge? J.L. Connecticut

ANSWER: No. In order to provide the Cartridge at a lower price for Schools and User Groups, we have made it available separate from its manuals. Anyone getting just one Cartridge should always get the Deluxe Cartridge Package. The books and disks included are a great help.

## COMAL 2.0 PRINTSCREEN NOTICE

I have observed a problem with the PRINTSCREEN procedure in the GRAPHICS package. The problem only shows up when 0 is used for the offset parameter. This causes the computer to send the decimal codes 27, 16, 0, 0 to the printer, which in my case does a reset of my Tymac Interface. Then while the interface is resetting itself, it isn't looking at the serial bus, and I get a "device not present" error (or something to that effect). Programmers might want to keep this in mind if they want their programs to work with as many systems as possible. M.L. Arizona

## COMSYMB FILE FOR COMAL CARTRIDGE

QUESTION: According to the COMAL HANDBOOK, Appendix P, machine language programs can be created using the Commodore Assembler Editor and a copy of the COMSYMB file on disk. How do I obtain a copy of this file? S.L. New Jersey

ANSWER: The COMSYMB file for the COMAL Cartridge will be included in the COMAL 2.0 PACKAGES book by Jesse Knight, available March 1985 for $19.95 (book and disk set).

## RELATIVE FILE BUG FIX

According to Jerry Claessen, the relative file problem on the 1541 is not completely solved in the 2.0 cartridge. [The problem is that COMAL gets ahead of the 1541 drive, causing information to be written to the wrong record.] He has had some problems there, but finds that setting the record delay up to 200 seems to fix everything.

## LEAVE YOUR CARTRIDGE PLUGGED IN

QUESTION: Do I have to unplug my COMAL Cartridge before using other programs such as PaperClip?

ANSWER: No. You can leave your cartridge plugged in at all times. To return the C64 computer back to 'normal' BASIC mode, simply issue the command: BASIC. To use other cartridges, of course, you will have to remove the COMAL Cartridge, unless you have a bus expander.

## LIGHT PEN IN COMAL 2.0 OFFSETS

I believe that the default values for the OFFSET procedure in the LIGHTPEN package are:

```
X correction =  75
Y correction = -45
```

The offset values are subtracted from the "raw" values from the pen before being passed back by the READPEN procedure. Also note that the time values in the DELAY and TIMEON procedures are in jiffies (1/60 sec). M.L. Arizona

## PROTECT PROGRAMS IN COMAL 2.0

QUESTION: What is the procedure for protecting programs as described on page 11 of COMAL TODAY #4? M.L. Arizona

ANSWER: Any COMAL 2.0 program can be protected so that it cannot be LISTED or MODIFIED - only RUN. Use the program called PROTECT64 on the Cartridge Demo Disk #3 to do this. Make sure you have another copy of the program - for once it is protected it cannot be UNPROTECTED! The disk is only $9.75 to subscribers. No one has reported breaking the protection yet. Get the disk and try it yourself!

## COMAL CARTRIDGE - A NON-PROBLEM

I finally figured out what the problem was with my cartridge - me! Can't use a system package name as a procedure name! Like some of you, I'd been using "sprites" as a procedure name without regard for the new "package". My cartridge seems to work fine now that I've stopped using reserved names. J.B.

## COMAL 2.0 ERROR HANDLER ADVANTAGE

QUESTION: When using relative records, is it possible to trap the error of reading a record that has not been written? It would be nice to display a message more meaningful than stack overflow. It took me quite a while to find the error the first time I got that message. R.C. Oregon.

ANSWER: Trapping an error is possible in COMAL 2.0, but not in 0.14. That is one of the major advances with the Cartridge. This is referred to as the ERROR HANDLER and is explained in the COMAL HANDBOOK.

## COMAL 2.0 PI AND TRACE

QUESTION: The following two keywords in COMAL 2.0 are missing from THE COMAL HANDBOOK: PI and TRACE. The format of the TRACE command seems to be:

TRACE [<filename>]

The screen is default. M.L. Arizona

ANSWER: Thanks for the tip on TRACE. We previously had been doing SELECT "LP:" followed by TRACE. Now we find it can be done all at once with TRACE "LP:".

## CAT & DIR - THE SAME THING IN 2.0

QUESTION: Why is there both a CAT and a DIR command? They appear to do the same thing.

ANSWER: Yes they both produce the same results. The only difference is that DIR may be used in a running program while CAT may not. Some computers use CAT for Catalog. Others use DIR for directory. To be compatible with both, UniCOMAL has included both names as commands. We use both IBM (DOS uses DIR) and Commodore (uses CATALOG) and like the fact that COMAL allows either.

## PASSING AN ARRAY AS A PARAMETER

In COMAL 2.0 the empty parentheses () are needed after the names of arrays being passed to procedures or functions. They MUST be omitted in COMAL 0.14.

## SHIFTED SPACE AND SHIFTED RETURN

Commodore introduced several peculiar things with their computers. For example, did you know that a shifted RETURN is different than the regular RETURN? In COMAL 0.14 you can use shifted RETURN to get to the next line without entering the current line. Also, a shifted SPACE is treated differently by Commodore computers (and also by COMAL 0.14). However, the COMAL 2.0 Cartridge corrects a few things including these. Now the RETURN key is the RETURN key; SPACE bar is SPACE bar, shifted or not. While it is nice to have these corrected back to 'normal', old habits will probably linger as you press SHIFT RETURN and then find the line IS accepted!

## C64 COMAL 2.0 ON DISK? NO WAY!

QUESTION: How much free memory is available after loading C64 COMAL version 2.0 from the disk supplied by Reston Publishing? Does the cartridge version of COMAL 2.0 have a lot more features than the disk? S.B. Connecticut

ANSWER: There is no such thing as a disk loaded COMAL 2.0 for the Commodore 64. What Reston has is the 2nd Edition of the disk to match the COMAL HANDBOOK, not COMAL 2.0. Their marketing misstated this (moved the 2 into the wrong place). The cartridge has LOTS of extra features, and 30K free memory for your programs.

## SETRECORDDELAY

This command in the SYSTEM package is designed to fix the 'bug' of the 1541 disk drive when doing RANDOM files (REL type files when cataloged). The problem developes when writing to a RANDOM file and the record being written falls on the break between disk sectors. The default setting is for the 1541 disk drive. If you have another disk drive type (MSD for instance) you can set the delay to 0 (SETRECORDDELAY(0) for example). This command only affects the speed that records are written to the disk drive, and only for RANDOM files. Remember, this command is part of the system package, so the command 'USE system' must be used before setrecorddelay is used.

## CHECK COMAL CARTRIDGE PROGRAM

If you think you have a bad COMAL 2.0 Cartridge, run the program below. It checks every bit in the cartridge, one bank at a time. There are four banks, so you should get four messages. If any of the four say CHECKSUM ERROR it is possible that your cartridge is bad. DOUBLE CHECK that you have typed in the program correctly.

```
DIM cart$ OF 1
USE system
PAGE
PRINT "comal cartridge checker"
PRINT "what color is your cartridge?"
REPEAT
  INPUT "(B=Black or G=Gray): ": cart$
  IF cart$ IN "Bb" THEN new'cart'test
  IF cart$ IN "Gg" THEN old'cart'test
UNTIL cart$ IN "BbGg"
//
PROC new'cart'test
  RESTORE new'cart
  test'it
ENDPROC new'cart'test
//
PROC old'cart'test
  RESTORE old'cart
  test'it
ENDPROC old'cart'test
//
PROC test'it
  all'ok:=TRUE
  FOR overlay:=0 TO 3 DO
    setpage(overlay+$80)
    PRINT AT 10+2*overlay,1: "overlay:";
    overlay;"==>"; // wrap line
    sum:=0
    FOR p:=$8000 TO $bfff DO sum:+PEEK(p)
    READ checksum
    IF checksum<>sum THEN
      PRINT "checksum error"
      PRINT checksum;"instead of";sum
      all'ok:=FALSE
    ELSE
      PRINT "ok"
    ENDIF
  ENDFOR overlay
  IF all'ok THEN PRINT AT 20,1: "your ca
  rtridge is ok!" // wrap line
ENDPROC test'it
//
new'cart:
DATA 1689037,1884343
DATA 1881580,1899936
old'cart:
DATA 1688895,1884630
DATA 1880026,1899936
```

## COMAL 2.0 PACKAGES

a book/disk set by Jesse Knight
$19.95 from COMAL Users Group, USA, Ltd.

Captain COMAL proudly announces THE book for serious 2.0 cartridge users. COMAL 2.0 Packages will show you the parts of and format for packages, as well as how to create and use your own packages. What this means to you as users of and programmers in COMAL, is that you can add anything you want to the language via machine language codings called packages. Customizations, enhancements, modifications to suit your whims or needs are all possible through the magic of packages.

It will also cover memory organization and control, how variables are stored, two's complement representation, floating point representation, parameter passing, and error reporting. There will also be a section documenting the system routines that can be called.

The disk which accompanies the book will contain an example module with the two packages, a package allowing the error messages to be changed or more added, and an MC monitor modified to work with COMAL 2.0.

## COMAL 2.0 AUTO ASCII CONVERSION

One of the many built in features of the COMAL Cartridge is TRUE ASCII conversion. Of this D.T. of Chicago writes: "I discovered, thanks to the hex-dump mode of my EPSON FX-80, that COMAL's ASCII conversion extends three characters beyond the alphabetic set: characters 91-93 are converted on a seven bit ASCII printer to the curly braces and straight line, but [, backslash, and ] are still available from the keyboard by typing SHIFT +, Commodore -, and SHIFT - (characters 219-221).

## COMAL COMMENT

The COMAL cartridge is the best improvement I've seen for the C-64. I've worked with Pascal (my first language), Fortran, BASIC, FORTH, and Assembler, but COMAL puts them all to shame with its power and ease of use. Congratulations! I hope COMAL replaces BASIC in the U.S.A. as it has abroad. T.D.

## POLAR ROSES

This program draws interesting loop patterns.

```
// delete"0:polar'roses"
// save  "0:polar'roses"
// by Captain Comal's Friends
//
WHILE NOT EOD DO
  READ p,q
  graph'rose(p,q)
  FOR delay:=1 TO 1000 DO NULL
ENDWHILE
END "End of program"
//
DATA 2,1,1,2,3,1,1,4,4,1,1,5,5
DATA 1,1,6,6,1,1,8,3,10
//
PROC graph'rose(p,q) CLOSED
  USE turtle
  hideturtle
  nowrap
  window(-190,190,-150,150)
  graphicscreen(0)
  clearscreen
  n:=p/q
  plottext(-190,140,STR$(n))
  b:=0
  sum:=p+q
  IF sum MOD 2 THEN
    maxval:=q*2*PI
    inc:=2*PI/100
  ELSE
    maxval:=q*PI
    inc:=2*PI/200
  ENDIF
  //
  WHILE b<=maxval DO
    r:=150*COS(n*b)
    x:=INT(r*COS(b))
    y:=INT(r*SIN(b))
    IF NOT b THEN moveto(x,y)
    drawto(x,y)
    b:+inc
  ENDWHILE
ENDPROC graph'rose
```

## SPRITE LOCATIONS IN COMAL CART

QUESTION: Is there any way to access the sprite data to change or view it? In other words, where in memory are they located? K.P. New York

ANSWER: In the COMAL Cartridge the sprite images are kept in RAM memory at $D000-$DFFF. To view this area you need to use the SETPAGE(0) command in the SYSTEMS package (images 0-31).

## COMAL 2.0 HINTS AND TIPS

from Kenneth Strahl

1-The EPROM board inside my cartridge came loose from the casing. My repairman informs me that this happens quite frequently in the HANDIC cases and should cause no problems with or damage to the cartridge.

2-I have been totally intrigued with the multi-colored graphics dump available on the cartridge. I am working on a program that places a light grey grid on a white screen and draws in dark grey and black, using the STAMPSPRITE command. While fooling with this I found a few quirks either in COMAL or the C64. Drawing and dumping 2 colors presents no problems. However, drawing and attempting to dump in 3 colors could result in the colors of the text screen bleeding through to the graphics screen, and seemingly turning off pixels on the graphics screen in some of the location where text appeared on the text screen. This occurs sometimes when accessing the printer with the PRINTSCREEN command, sometimes when using the SAVESCREEN command, and sometimes simply when flipping back and forth from text to graphics screens. Even though some pixels appear to be off at this time, the computer still considers them to be on, so that attempting to draw within a few pixels of these areas will make the "erased" area re-appear again. I have minimized this problem by flipping back and forth from the graphics to the text screens a few times before beginning to draw, and avoiding, whenever possible, drawing so that all three colors overlap. While drawing in this manner, I also flip to the textscreen occasionally and see if any damage has occured. If it has, any of the "erased" areas can be drawn a second time. They are usually not erased after the second time.

## CASE LOCK & UNLOCK

In order to "lock" your system into whatever the current case is (either upper-lower, or upper-graphic), all you need to do is press Control "H". Once the lock is in effect, pressing the Commodore/Shift combination will not have any effect. To release the lock, and re-enable Commodore/Shift, press Control "I". This works for both 0.14 and 2.0.

## AVE ATQUE VALE

Colin Thompson, who has done such a superlative job of collecting, assembling, and editing the user group disks, stunned us recently by announcing his retirement from this form of creative endeavor. With 8 disks tucked under his belt, Colin decided it was time to turn his talents to other "projects". With a tear in our eye, we say "Thank you Colin...and happy trails to you."

Does this mean the User Group Disks are a thing of the past? Was all Colin's work in vain? No, no, we cry! Such a thing cannot be!

Seriously, the User Group Disk tradition continues, albeit on another coast. Disks can now be sent to Mindy Skelton, PO Box 76, Mt. Holly Springs, PA 17065. For those of you with access to PlayNET (and that is many of you, I hope), files can be uploaded to Mindy via PlayNET's file transferal capabilities. If you decide to upload the files to Mindy S (as she would encourage you to do), please make sure you remove all occurences of an ' in the file names, as this symbol, for some reason, causes the PlayNET system to refuse your file.

## IMPROVED FILE'EXISTS FUNCTION

by Len Lindsay

In the COMAL handbook, page 381, a sample function for COMAL 0.14 called "file'exists" is listed. The sample indicates that you should check for a value of "62" to determine whether or not the file exists. Even though the function, as written, performs correctly, I am providing an improved version. The improved version also checks for such problems as open drive door, read errors, or any disk problem.

```
FUNC file'exists(file'name$) CLOSED
  DIM s$ OF 2
  OPEN FILE 78,file'name$,READ
  s$:=status$ //just first 2 characters
  CLOSE FILE 78
  IF s$<>"00" THEN
    RETURN FALSE
  ELSE
    RETURN TRUE
  ENDIF
ENDFUNC file'exists
```

## FILL KEYBOARD BUFFER

Here is a procedure that will fill the keyboard buffer with the string you pass to it. Remember that the buffer can only hold 10 keystrokes.

```
PROC buffer(string$) CLOSED
  l:=LEN(string$) MOD 11
  FOR x:=1 TO l DO
    POKE 630+x,ORD(string$(x))
  ENDFOR x
  POKE 198,l
ENDPROC buffer
```

## SEVERAL FILES OPEN AT ONCE

QUESTION: Will COMAL read and write to more than one file concurrently? If so HOW and HOW MANY? J.H. Missouri.

ANSWER: Yes. Our order processing program (written in COMAL 2.0 of course) has 4 files open all the time. The disk drive takes care of all the work. That is why the number of files depends upon the disk drive you use. We use a CBM 8250. When we transferred our orders program to IBM COMAL, we had to change it to have one less file open.

## DISTRIBUTE THE COMAL 0.14 PROGRAM

QUESTION: We are writing COMAL 0.14 programs that we hope to sell. May we include the COMAL 0.14 system on the disk with our programs?

ANSWER: Yes. You have permission to include the COMAL 0.14 system unchanged on your disk as long as the copyright notice is included on your disk label. Also, you must send us one copy of each disk you sell for our records. However, no royalties need to be paid. Keep in mind that you may sell your program and include the COMAL 0.14 system free. You may not sell COMAL 0.14 itself without our prior approval.

## NEW COMMAND MESSES UP GRAPHICS

Phyrne Bacon reports that the NEW command messes up SETGRAPHIC. She found that DEL 10- works fine, and allows you to then ENTER and RUN a program such as SAVE HIRES COLOR.

## LIST TO DISK WITH INDENTATION

If you LIST a program to disk in COMAL 2.0, it will include the indentations, while in COMAL 0.14 it did not. To get a program listed to disk in COMAL 0.14 with indentations just do this:

```
OPEN FILE 255,"0:NAME.L",UNIT 8,WRITE
SELECT "LP:"
LIST
```

That is the way to do it for NEWSLETTERS and MAGAZINES. They can easily chop of the line numbers - especially in PaperClip, which has a delete column command that does this perfectly.

## TWO COLUMN DIRECTORY

by Ray Carter

You have probably at one time or another used the PRINT'DIRECTORY program from the 0.14 sampler disk. My main complaint about the program as it was distributed is that it puts the directory on the left half of the paper, and leaves the right half blank and wasted. With the price of paper being what it is, I decided it would be nice to print the directory in two columns. I also decided that it might be a little more convenient if the directory listing read from top to bottom down the left column, and then down the right column rather than jumping back and forth (that, of course, would have been the easy way to do it). At any rate, the enclosed program will do that for you. Another nice touch is that the left column is moved in a few spaces to permit sticking several directories in a folder for permanent reference.
This program is on TODAY DISK #6.

## REPEATING KEYS

(Hold a key down and it repeats)

```
PROC repeat'keys(on'off)
  IF on'off THEN
    POKE 650,128
  ELSE
    POKE 650,0
  ENDIF
ENDPROC repeat'keys

repeat'keys(TRUE) // turn on repeat keys
repeat'keys(FALSE)//turn off repeat keys
```

## FASTER RAM ERRORS SYSTEM

by Phyrne Bacon

In COMAL TODAY #4, Glen Colbert introduced a system to have the error messages in RAM memory, thus preventing disk access every time an error is made. His system used the original COMALERRORS file on the disk and copied it byte by byte into RAM. To speed up this process involved creating a new error message file from the original. The program GENERATE'NEWERRS takes care of this, creating the new error message file named "---ERROR-MESS---". The routine ERRORS'TO'RAM quickly loads the error messages from disk into RAM. It is part of the new HI program on TODAY DISK #6.

## 1541 FLASH AND COMAL

A.T. in Massachusetts informs us that the COMAL 2.0 Cartridge works well with the Skyles' 1541 Flash. He used the PRESENTATION program on Cartridge Demo Disk #1 for timing tests with the following results:

| | Normal | Flash |
|---|---|---|
| LOAD | 62 | 23 |
| SAVE | 64 | 52 |
| LIST | 100 | 83 |
| ENTER | 207 | 139 |

## TURBO 64 AND COMAL 0.14

I have found that TURBO 64 from MegaSoft reduces the loading time of COMAL 0.14 from 87 seconds to 17 seconds. This $20 program consists of two ML programs and loads in under 9 seconds, so the savings is significant. Can anyone supply a COMAL BOOT program that first loads TURBO 64 and then COMAL 0.14. Of course TURBO 64 would not be included on the COMAL disk - but could be copied to the disk by the user. J.V. Massachusetts.

ANSWER: If someone has such a BOOT program, we will be happy to distribute it on the next COMAL TODAY disk or a User Group Disk.

## COMAL 0.14 LOWER CASE PLOTTEXT

To have PLOTTEXT plot in lower case simply issue the following command:

```
POKE 29260,216
```

## LOAD MISTAKE TIP

COMAL 0.14 does not like it when you try to LOAD a BASIC program (or a picture file or wordpro file either). But BASIC is equally upset when you try to LOAD a COMAL program (in fact trying to LOAD a COMAL 2.0 program into BASIC will totally crash the computer). How can you tell if the program you loaded was really a BASIC program? Simply say: LIST. If nothing is listed, then it was not a good COMAL program. To recover, simply issue the command: NEW. (Don't issue the RUN command or you WILL crash the system).

## LOAD BUG

COMAL 0.14 seems to have a bug in the LOAD statement. When you have a large program in memory, and attempt to LOAD in another large program (the total size of the two programs exceed memory), COMAL will stop and report an error. What happens is that COMAL does not reset certain memory pointers, and a memory overflow occurs. This problem is easily avoided by using the NEW command before you LOAD a large program. The CHAIN command does not seem to have this problem.

## COMAL HI RES PICTURES

QUESTION: I am writing a HI RES drawing program in COMAL 0.14 and would like to SAVE and LOAD a picture, as well as print it on a printer. Any help?

ANSWER: Several HI RES dump programs are already available (on TODAY DISK #3, #4, #5, and UTILITY DISK). A picture LOAD / SAVE routine is on TODAY DISK #4 and #5. COMAL TODAY newsletter explains these routines.

## DIRECTORIES OF DISKS

QUESTION: Please send me information about what is on EACH of the User Group Disks and EACH of the TODAY DISKS. I am going to buy several, but need to know what is on each one. T.L. Texas

ANSWER: The disk directories of 29 of our COMAL disks is printed in COMAL TODAY issue #5.

## 19 DIFFERENT COMAL DISKS SET

QUESTION: I heard that it is possible to get about 1000 COMAL programs on 19 different COMAL disks for less than $100. That is less than 10 cents per program. Is this true? Can they be copied for friends? Can they be included in our clubs disk library?

ANSWER: Yes to all those questions (alot of them too). Just ask for the 19 DISK SET for $94.05 to any COMAL TODAY subscriber.

## WHAT IS THE COMAL STARTER KIT

To make it easy to get and affordable to get started with COMAL we have put together a package we call the COMAL STARTER KIT. It includes the book COMAL FROM A TO Z and three disks: TUTORIAL DISK, AUTO RUN DEMO DISK, and SAMPLER DISK. It comes in a custom molded plastic case, similar to those used by SPINAKER.

## BOOK REVIEWS AND COMPARISONS

Next issue we hope to print a review of EVERY COMAL book currently available, and if possible include comparisons between books. We welcome your contributions! If you have one or more of the COMAL books send us your comments and recommendations.

## ROUND - USER DEFINED FUNCTION

QUESTION: Have I discovered a missing command in the COMAL Cartridge? The ROUND function? When I try to use ROUND I get an error: ROUND: UNKNOWN ARRAY OR FUNCTION. K.S. New York

ANSWER: ROUND is not part of the COMAL KERNAL (Standard). It is added as an enhancement to Metanic COMAL (CP/M) and IBM PC COMAL. Commodore COMAL does not include it, and it is not listed in the COMAL HANDBOOK as a keyword, but is included in APPENDIX D, page 395, as a user defined function. Here is how you could do it:

```
FUNC round(number) CLOSED
 RETURN SGN(number)*INT(ABS(number)+.5)
ENDFUNC round
```

## PINWHEEL

by Jennifer Minge

Here is a short program to draw a pinwheel design.

```
// delete "0:pinwheel"
//  by jennifer minge
// save   "0:pinwheel"
background 0
setgraphic 0
border 0
pencolor 7
hideturtle
fullscreen
count:=1
this'count:=1
while count<=36 do
 forward 75
 while this'count<=8 do
  forward 5
  right 45
  this'count:=this'count+1
 endwhile
 back 75
 right 10
 count:=count+1
 this'count:=1
endwhile
```

## PERFECT MATCH FOR THE TUTORIAL DISK

By now most of you already have the TUTORIAL DISK. Gordon Shigley designed it to be an interactive set of lessons about COMAL. He also wrote the COMAL WORKBOOK as its companion. Together, the TUTORIAL DISK and COMAL WORKBOOK make a good introduction to COMAL, including graphics. The WORKBOOK is now in stock for immediate delivery. Quantity discounts are available for schools and user groups.

## COMAL SUPPORT NEWS

by Captain COMAL

NEWS ONE: There now is a bulletin board service for COMAL users.

NEWS TWO: There now is a multiple user online system allowing up to twelve COMAL people from around the country to 'talk' to each other.

NEWS THREE: There now is an easy way to transfer your new COMAL programs to the COMAL Users Group - and to get a program back in return - without leaving your house.

The good news is that PlayNET is supporting COMAL. Don't mind the word PLAY in their name. This is a serious and comprehensive national multi user network system. Through PlayNET you can now have access to a COMAL Bulletin Board System, Electronic File Transfer, and Online Multi User Communications. And the best news is that it only costs $2 an hour to use this service (plus $6 per month account fee).

PlayNET and COMAL have alot in common, even in their names, which seem to turnoff alot of people (COMAL sounds too much like COBOL and PlayNET sounds like only fun and games). Another thing in common is that once people try it, they don't just like it, they **LIKE** it. Both PlayNET and COMAL are one step ahead of the 'others' and try to make best use of the computer. Both are relatively new in their area - and both tend to be ignored by the magazines. Both are designed for 'the people' and priced accordingly. Finally, both are using the Commodore 64, the 'peoples computer' as the base to grow from. So, it is only natural, that COMAL use PlayNET to advance communications between its users.

PlayNET has supplied a special section in their Bulletin Board area JUST FOR COMAL. This section is called COMAL (of course) and is maintained by the COMAL Users Group USA. Watch it for late breaking COMAL news and information.

You also can use the Electric File Transfer service of PlayNET to send your COMAL programs to the COMAL Users Group. Just UPLOAD the program and leave EMAIL to CAPTAIN C with the name of the file and who it is from. PlayNET does NOT charge you to do this. In return, I will try to put one COMAL program up each day for you to download. PlayNET will charge you 50 cents to download the program.

Finally, you can now get your COMAL questions answered instantly. Every Thursday night, 10pm until Midnight E.S.T., you are invited to the COMAL room on PlayNET where Colin Thompson and David Stidolph assist me in providing answers. And if we don't know the answer, maybe another person online does. There are already dozens of COMALites using PlayNET, and the COMAL service hasn't even been officially announced yet (word of mouth travels fast). There should be about 400 COMALites on PlayNET before the next issue of COMAL TODAY is printed. If you have a modem, can you afford to miss the chance to talk with other COMALites around the country any night of the week and all day during the weekend?

The only hitch is that PlayNET charges $39.95 as an enrollment fee when you sign up. But fear not, PlayNET has agreed to waive this fee for COMALites - for a limited time - if COMAL Users Group would provide each new PlayNET subscriber with the PlayNET system disk and manual. That means it cost you nothing to sign up - once you have the system disk and manual. And - yes, you guessed it - we are even making it possible for you to get the disk and manual free as well (see special offer elsewhere in this issue).

When you join PlayNET, remember to send me EMAIL with your PlayNET name. I will try to keep a logbook of all COMALites using PlayNET.

And, since PlayNET is something that most of you have never seen, we have included a PlayNET simulation on TODAY DISK #6. You don't need a modem to try the program. It is a regular COMAL 0.14 program. Just LOAD and RUN it. Then you will know what PlayNET looks like.

## COMAL TODAY EUROPEAN DISTRIBUTOR

It looks like all European COMALites can now get COMAL TODAY direct from:

Nils Hansen
Post Box 1222
DK 2300, Copenhagen-S
Denmark

# DATA BASE MANAGER IN COMAL 0.14

by Robert Shingledecker and
Steve Smullen

ED NOTE: We were all waiting in anticipation for a DataBase written in COMAL. Then they appeared - not one but two different Data Base systems. This one is a general purpose data base, useful for a variety of things. It is included on TODAY DISK #6. Elsewhere in this issue you can read about the other data base by Will Bow, specifically for cataloging a disk library.

**OPERATING INSTRUCTIONS**

DISK PREPARATION:

Using standard procedures FORMAT a disk for use as a data base disk. Next copy "data'base'mgr" and "dbase14" onto this disk.

STARTING THE PROGRAM:

With the data base disk described above mounted into the disk drive type CHAIN "DATA'BASE'MGR" press return. Wait for the menu to appear.

f1 - Create a data base
f3 - Use a data base
f5 - Purge a data base
f7 - List all data bases on this disk
f8 - End program

CREATE A DATA BASE:

Before you begin this option you should plan ahead. Think about what you want to capture. Think about how the data will have to be sorted. Sorting is on a field level. You can have up to twenty fields per record. Records may contain up to 254 characters. The field name plus the field length must be able to be displayed on one line, i.e. maximum 40 characters. Pre-planning will yield the most useful dictionary thus a more useful data base.

The program will prompt for field name. This is not to be confused with the data. This is the PROMPT name for this field. What you type here will be used by the "USE" option elsewhere in the program. Try to keep prompt names short. Remember the prompt name and the field length must fit in 40 columns.

The program will next prompt for the field length. If the name and field length are more than 40 characters then break it up into more than one field.

Next the field type will be prompted. Type an "A" for alphabetic only, or an "N" for numeric only, or (best choice) type a "B" for both. Only use the "A" or "N" when you really want to restrict the input to a certain type!

The program will now display the prompt together with the input field based on your specifications. Enter a "y" if it is correct or a "n" will allow you to restart this field definition.

The program waits for an entry of "c" to continue field definitions or an "e" to end field defintions. If you select "c" continue entering field definitions as just described.

Next enter the name of this data base. Do not use "'dict" or "'data" as these are reserved suffixes used by the program. Use simple one word names!

Now enter the approximate size of the data base. That is how many records. Be realistic! This is NOT an IBM mainframe! Keep the size small, usually less than 100 records works best. The system will handle up to 400 records but will take a VERY LONG time to sort! The limit of 400 records is based on the number of free blocks with a maximum record size.

Now be patient as the system will set up the disk files for the dictionary and the data portions of your data base.

USE DATA BASE:

Use this option to enter new records into the data base. You will be prompted with the field names that you created. The field length will be displayed and the keyboard input will be restricted all based on your definitions. Use the delete key to correct mistakes within a field. After all fields have been entered you will be prompted to accept the record or re-input the entire record. If you are entering a record with many fields and discover an error it might be best to accept this record and change it later with the "change" command. The "change" command allows easy changes by field.

Finally you will be prompted to continue adding more records or return to the use option menu.

CHANGE RECORDS:

Use this option to change any or all fields of an existing record. Once you select this option you will be prompted for the select method desired. See "Select Options Menu". Once a select option is chosen the system will try to find the desired record(s). Once found the record contents will be displayed. Enter a y to change the record. Then for each field the prompt and input field will be displayed. If you DO NOT want to change this particular field simply press RETURN key. The original field contents will appear and the next field prompt and input field will appear. By simply pressing the RETURN key you can "tab" to the desired field to change. When all fields have been displayed the program will respond with "Change record? (y,n)" for a final approval. If you select "n" you can still recall the original field contents. When you select "y" all fields will be updated. Note anytime you change records you must re-sort the file to use the quick find select option!

DELETE RECORDS:

This option will delete records from the data base. Once this option is selected you will be prompted for the select option. For each record that matches the select option chosen that record will be displayed and a final prompt to delete it will be given. Note any time you delete records you must re-sort the file to re-gain the use of the quick find select option.

DISPLAY RECORDS:

Use this together with the select options for a powerful combination of displaying information. Simply press the SPACEBAR to view each record that matches the chosen select option.

PRINT RECORDS:

The same powerful capabilities for displaying information are also available while printing!

SORT RECORDS:

With this option all field prompt names are displayed. Simply enter the field name you wish to sort by. Be PATIENT! The Commodore serial bus simply cannot drive the disk very fast. This process is VERY SLOW so just sit back and relax! Wait until you see the "Use Options Menu" to re-appear!

SELECT OPTIONS MENU

f1 - Complete Search
f3 - Quick Find Record
f5 - Process All Records
f7 - Return to Use Options

COMPLETE SEARCH:

Enter string searching for: [ ..] will appear. Enter the characters that should reside in the records you desire. The less you specify the more records that are likely to contain them and thus be selected. The more characters you specify the less records will be selected. Each record that contains the specified string regardless in what field will be selected for processing by the previously specified mode, i.e., changing, deleting, displaying, or printing.

QUICK FIND RECORD:

This option is only available when the file is sorted. Once selected the program will display the sort status of the file. If the file is sorted, the field in which it is sorted by, will be displayed. You will be prompted if this is the field that wish to select by. Enter the EXACT field contents for the record that you want to locate. If the record is found then the record will be selected for further processing by the current mode.

PROCESS ALL RECORDS:

This option will select ALL records in the file for whatever mode of operation was chosen, changing, deleting, displaying, or printing.

RETURN TO MAIN MENU:

f6 from the Use Options Menu will return to the Main Menu.

PURGE A DATA BASE:

Enter the data base name to be removed from the system.

LIST ALL DATA BASES:

This option will display on the screen all data bases residing on the disk.

## QUOTE MODE CONTROL CHARACTERS

QUESTION: I have the COMAL 2.0 Cartridge and am having difficulty with the CURSOR LEFT, CURSOR RIGHT, HOME, and CLEAR SCREEN in the quotes after the print statement. The text says that COMAL will put the ASCII codes in for you. This does not happen. If I use a CURSOR LEFT the CURSOR moes to the left. WHat am I doing wrong? The only way this works is if I put the number in double quotes. M.C. New York.

ANSWER: Any program ALREADY WRITTEN (in COMAL 0.14 for example) will automatically have the CURSOR symbols converted into their equivalent ASCII control codes. The COMAL cartridge defaults to QUOTE MODE OFF. COMAL 0.14 always has QUOTE MODE on. If you like QUOTE MODE, just issue this command:

```
USE SYSTEM
QUOTE'MODE(TRUE)
```

Now, you can put cursor movements into strings as you are used to. Then issue the command: QUOTE'MODE(FALSE) and see all those converted into control codes. For example, the following lines show the control codes as with QUOTE'MODE off (the default in the cartridge):

```
PRINT ""19"", // home cursor
PRINT "PRESS "18"RETURN"146" TO CONTINUE"
PRINT ""17"HELLO"
```

A control code needs a set of quotes around it. Thus if it is the only thing in the string, you will find double quotes around it (first example). However, if it is inside a string constant, you will only find a single quote on each side (second example). And of course, you will find control codes at the start or end of string constants as well (third example).

## PRINTERS AND COMAL

QUESTION: I am having a problem with some routines which call for a printer printout. Many of the programs work OK, but a few are giving me some trouble. Can you offer some suggestions? J.W. Arizona.

ANSWER: A common request is how to access the secondary address of the printer. Anything sent to the printer from COMAL 2.0 is default in 'lower case' mode, while COMAL 0.14 defaults to the UPPER CASE/ graphics characters set. Many printer interfaces utilize the secondary address to flip between various 'modes'. Here is how to access this:

```
COMAL 0.14:
OPEN FILE 255,"",UNIT 4,7,WRITE
SELECT "LP:"

COMAL 2.0:
SELECT "LP:/S7"
```

In COMAL 0.14, you must use file number 255. If your printer is device number 5, simply replace the 4 with a 5. In COMAL 2.0, simply include an 'attribute' specifying secondary address: /S.

Also ESCAPE CODES are often used to control various print modes in many printers. You can send these from COMAL:

```
PRINT CHR$(27)+"G",
```

This will send an escape G to the printer IF a SELECT "LP:" is in effect. The comma at the lines end suppresses a carriage return. If you would like to OPEN the printer AND send it an ESCAPE CODE at the same time you can make the escape code the 'filename':

```
COMAL 0.14:
OPEN FILE 255,CHR$(27)+"G",UNIT 4,6,WRITE

COMAL 2.0:
SELECT "LP:"27"G"      or
SELECT "LP:"+CHR$(27)+"G"
```

We tested it on a straight thru parallel port (on a BUSCARD) and found it works fine with a CENTRONICS GLP printer. However, going through some interfaces, it doesn't seem to work, the interface is interfereing somehow.

# DISTRIBUTION OF DISK LIBRARY LIST

A GENERAL PROBLEM - by C64 WEST
by Will Bow

ED NOTE: This fantastic DISK DATA BASE system is extremely easy to use - no manual necessary. We liked it so much, that we are using it to list all the COMAL disks we have available. You can get a copy for yourself right now. The data base complete with all 29 COMAL disks data is on the back side of the BEST OF COMAL disk - only $9.75 to COMAL TODAY subscribers. Now, Will Bow explains how he created this masterpiece.

Most User Groups have a library of disks available for copying by members. As the number of disks grow, how do you let the members know everything that is available?

A number of approaches to the problem were considered and abandoned. The idea of a printed list lost out early because it cost too much and couldn't be updated easily. Most of the other approaches considered also lost out for similar reasons. Finally, the computer was looked to as a solution to the problem (kind of poetically appropos, after all, we are a computer club aren't we?). A database program was the natural answer. It became evident that we would have to write our own database program because no assumption could be made that the entire membership would have access to any one particular commercial database program.

Systems considerations and methods:

1- System limits. The specifications as to maximum number of records that the database could handle as well as the size of any given field were determined by the limitations of the hardware. The system uses a 'floating' record size that will be explained later. For the purpose of this section, a definition of the term 'record' is given here. A record is defined as one complete disk directory listing with comments if any. It's length may be anywhere from 17 bytes long to a staggering 5328 bytes long! The 'floating' record size is accomplished by using 'pseudo 'sectors' in the database. Each 'pseudo-sector' is 32 bytes long and may contain either a filename and filetype descriptor or a comment. As many of these 'pseudo-sectors' are chained together as necessary to achieve the needed record length. A total of 4000 'pseudo'sectors' are available. The total number of disks that may be cataloged is dependant on the number of files and the number of comments associated with the disks that are already cataloged. Therefore, its really hard to say exactly how many disks may be cataloged to the database. However, the system does have some fixed specifications.

The system allows you to have 'sublibraries' -- that is, you may segragate your disks into catagories like UTILITIES, APPLICATIONS, COPIERS or whatever else you want. This allows you to browse a specific 'sublibrary' for what you wanted instead of having to wade through an unorganized mass of disks. The system is set-up to allow a maximum of 9 'sub'libraries' with a maximum of 36 disks per library. If a given 'sub'library' goes beyond the 36 disk limit, you may start another 'sublibrary' and call it something like 'UTILITIES II' as opposed to 'UTILITIES I.

2- Record Size. As mentioned above, a record is defined as one complete disk directory listing with comments if any. The minimum size being seventeen bytes (sixteen bytes for the filename and one byte for the filetype descriptor i.e. "P" for prg files, "S" for seq files etc.). The maximum size being a staggering 5328 bytes (144 times the above + 144 times the length of the comment line (20 bytes) and 144 bytes more for filetype descriptors) assuming, of course, that each filename carried with it a comment line). No commercial product exists that will even begin to handle that kind of load. Now, keep in mind that the program must be able to summon up anywhere from 17 to 5328 specific bytes of information on the demand of about 5 keystrokes and you get some idea of the problem involved in writing a system of this type.

3- Memory Size. COMAL 0.14 was the language of choice for this project. It was widely available to all at no cost and there was no way in the world that I was going to try to write a program system of this magnitude in BASIC (as you can see -- I am no fool!). Default memory size for programs in COMAL 0.14 is 9902 bytes. This was insufficient for the task at hand because, even with program

chaining techniques employed, some of the modules exceeded 11K bytes in length! Well, thanks to John Mc Coy in Texas, COMAL 0.14 has been expanded to 11,838 bytes. Memory is expanded in the first module called "HI" and stays expanded throughout the entire run of the system. The proc, called 'EXPAND'COMAL', may be extracted and used freely in your own programs.

4- 'Dynamic Menus' and Logic structures. Most programs that use menus use a 'Fixed' menu system -- That is, they are 'Hard Written' routines that offer not only a fixed list of items to choose from but also a fixed number of items. This is fine in most cases but something different was needed for this database system. The menu system used here is 'dynamic' -- That is, depending on what has been cataloged into the system, you are not only offered a different list of items to choose from but a variable number of them as well. 'Paging' is necessary in some cases. No problem -- it's all automatic and carried on by the logic structuring used.

If you analyzed a COMAL flowchart, specifically it's 'Decision Making' structures, you will notice a definite 'cascading' effect. Again, fine in most cases but in situations where menus call sub-menus which in turn call sub-sub-menus where each is dynamic and may involve 'paging' where the program flow must return to a given page, 'cascade' structures are inadequate. 'Branching Tree' or 'Nodal' systems like those used in FORTH are more appropriate.

5- The BAM (Block Availability Map). The database file itself is made up of some 4000 individual 'pseudo-sectors' which are chained together to meet the dynamic range requirements of the individual records (remember, a record may be anywhere between 17 bytes to 5328 bytes long). Some method had to be devised to know which 'pseudo-sectors' were already used and which ones were not. The BAM, not unlike the BAM used by the CBM disk drive, serves this purpose. In essence, it is a 4000 bit binary switch. It is only used by modules that affect the database itself i.e. 'add'disk' and 'annotate'disk'. Physically, it occupies 500 bytes of memory starting at $c000. It is interrogated from left to right with an 'in-use' status represented by a 1 bit and conversely, 'free' status represented by a 0 bit. The logic used to set and un-set particular bits in the switch is straight boolean logic, AND, OR and XOR and may be found in the 'logical'ops' demo. The functions mentioned above depend on this information when it is either looking for free space or letting us know that previously used space has been freed and is available for use. Flexfile and just about every other commercial database software use a similar method. Flexfile refers to it as 'bookkeeping information'. They warn that this information is vital to the database's functioning and that it is imperative that the software be shut-down properly so that this information is saved. This database is no different. In the 'Add' and 'Annotate' modes you will notice screen information telling the you that the BAM is either being LOADed or SAVEd. This is done at the start-up and shut-down of those particular functions and and must occur as part of their running sequence else the whole database is lost and rendered useless. There are usually no problems as long as the program is allowed to run to it's natural end as the LOADing and SAVing routines are automatic. Any attempt to circumvent this chain of events could result in havoc. So, whatever you do, DO NOT hit RUN STOP/RESTORE while in those modules. Instead, follow the screen prompts and let the shut-down sequence follow its natural course.

6- Record Chaining. Because of the variable record lengths involved, a method of 'dynamic record storage' had to be devised. Record chaining was the answer. As you catalog disks into the system you will notice a growing list of filenames on the library disk itself that reflect the names you used to create the individual sublibraries. These are the index files used to locate the desired directory list in the relative file database. For every disk cataloged to a given sublibrary there exists an entry in the index file that includes the disk name, disk i.d., and the first record in the chain of individual filenames which make up that record. Each of these records in the database contains a chaining field which point to the next record in the chain. The process continues until the end of the chain is

reached (9999 in the chaining field). So, there we have it -- a way of storing all those file names whether there be one or 144 using as little disk space as possible. Now, the comment records. The comment records constitute a 'sidechain' to the main chaining structure.

7- Table Management. To facilitate rapid 'paging' and also get around the problem of 'reverse record chaining', in the view disk mode the entire record is memory resident. That's right, all 5328 bytes of it! Due to the memory size limitation and the amount of memory overhead involved in conventional table processing, the use of conventional tables was impossible. 'Pseudo tables' had to be devised -- a method of substring extraction involving cancantating fixed length strings into one huge string and and accessing any given substring with the formula:
((((index-1)*rec'length)+1) :
(((index-1)*rec'length)+rec'length))

8- Screen Management. Screen management is purely a matter of asthetics. Designing these 'User Interfaces' is not my speciality. I needed help. The screens employed here were designed jointly by Colin Thompson and myself. I seeked out Colin Thompson, who is good at that kind of thing, to ask his advice. Colin wanted to see the 'sliding bar' method of menu selection. He said that that would give it 'a touch of class'. I responded, 'What are you, nuts or something? Do you have any idea what that is going to take??'. I thought about it for a while and sat down to try it and what you see is the end result. Colin was right, it did look real good. But then again, I didn't doubt that it wouldn't; Colin is good at that kind of stuff. No real magic involved. The techniques are straight forward and easy to analyze on your own. I threw this in to say that COMAL itself is still a very young language and few people have mastered it fully. However, a real network of resource people who have 'chopped big holes in the woods' in some particular aspect of COMAL is beginning to form and those people are worth consulting when the problem lapses out of your area of experience. It would be worthwhile for you to create similar networks of resource people. Many thanks to Colin for his help and constructive criticisim in this project.

Well, that's just about it. You should have enough information here to write similar database systems for your own use. Please feel free to write to me. Will Bow, 1531 Corinth Av. Apt. 9, West Los Angeles, Ca. 90025. My specialities are array handling and relative file management.

## COMAL 2.0 BATCH FILES

by Captain COMAL

A batch file is like having a keyboard buffer the size of a floppy disk file. Indeed, that is what it is. It's uses may not be apparent to you right now, but sooner or later, the batch file will save your day.

A batch file is merely a SEQ file of ASCII COMAL commands and text. It can be created by any means you wish (even PaperClip will work). A batch file EDITOR is included on Cartridge Demo Disk #3. Or just use COMAL PRINT FILE statements to create your own personalized batch file. For instance, let's create a batch file that will first provide us a CATalogue of the current disk, and then give us the SIZE of the program in memory:

```
OPEN FILE 2,"BATCH'TEST",WRITE
PRINT FILE 2: "CAT"
PRINT FILE 2: "SIZE"
CLOSE
```

That created your batch file. This is how to get it started:

```
SELECT INPUT "BATCH'TEST"
```

Control is now transfered to the batch file, and COMAL treats all characters coming from the file as if they were typed on the keyboard. Our example will now do a CAT followed by a SIZE. To see batch files in use refer to programs VIEW'FONTS (Cartridge Demo Disk #3) and BATCH'COPIER (Cartridge Demo Disk #2). You may wish to create a batch file that you call when you start working - one that sets up your screen colors and defines all the function keys the way that YOU want them. If you come up with some nice batch files, we hope you will share them with other COMAL TODAY readers.

# COMAL IS COMPATIBLE

by Len Lindsay

One reason that COMAL is here to stay is that a COMAL STANDARD exists, called the COMAL KERNAL. About twice a year all the people involved in producing COMAL systems meet and try to keep all the versions of COMAL compatible.

It's nice that all the COMAL implementors talk with each other. And it is nice that they all intend to keep their versions compatible. It's nice in theory. But are they compatible in the real world?

We have talked alot about COMAL 0.14 and COMAL 2.0 - usually referring to versions for the Commodore 64. Now, we also have COMAL 2.0 for the IBM PC. And not just one, but two different implementations. First we tried all the programs from the COMAL HANDBOOK (which was written for COMAL on Commodore computers) on each of the two IBM PC COMALs. Amazing. Almost everything worked. Only a few very minor differences.

But I wanted a real test. What better test than to take a LARGE program written for Commodore COMAL and run it under IBM PC COMAL? We have a perfect program for this - our order processing system. It is so large that we can't number the lines by 10 - we must number it's lines by 5. OK, so we have the program. Now the trick was getting it into the IBM PC - the disk format is incompatible with Commodore's disk format. We did NOT want to retype the whole program.

Aha! The answer was simple. We just ran a terminal program on both computers, and transfered the program over the phone via modems. It worked the first time. We now had a duplicate copy of the ORDERS program on our IBM PC disk. Now the test.

We booted up IBM PC COMAL 2.0. Then we issued the command:

```
ENTER "ORDERS"
```

I was shocked. IBM COMAL had problems with only 2 lines. And those two lines were to disable and then enable the stop key. That feature was not implemented yet (we have a preliminary IBM PC COMAL), but will be implemented in the final. So, with the final version of IBM PC COMAL, the entire file would have been entered without a hitch.

But that is only syntax compatible. How about actually running it? First we had to change the file names. Those will have to vary to match the operating system of the computer. Commodore has a file name of up to 16 characters. IBM allows up to 8 characters plus a 3 character extension. We used 6 files in the program, so each file name had to be changed. Also, at the same time note that Commodore refers to its drives as numbers (0:, 1:, etc.) while IBM refers to its drives as letters (A:, B:, etc.). If you use the default drive, this is no problem. But we had specified drive numbers. Another easy change - we just changed the drive numbers 0 and 1 into letters A and B. That was it!

Then we tried running the program. Another difference now popped up - also dependent upon the computers operating system. IBM COMAL would not let us have all our files open at once. It allowed only 4 open files (counting the printer as one of them), while Commodore allowed us 4 not counting the printer. For most of you, three simultaneously open files plus a printer should be sufficient. We changed one of our files so it could be closed most of the time - and only opened it while the printer was not selected. Also note that IBM COMAL only allows file numbers 1 through 20. This was no problem for our program as we used small file numbers.

We hit one more 'snag'. That was only because we were using a 'shortcut' in our substring use. A substring in COMAL is referred to like this:

```
TEXT$(start:end)
```

To take substring of characters 3, 4, and 5 we would say:

```
TEXT$(3:5)
```

If we only want the third character we are supposed to say:

```
TEXT$(3:3)
```

That will work in both Commodore and IBM COMAL.

However, Commodore COMAL allows this short cut:

```
TEXT$(3)
```

while IBM COMAL allows a different short cut:

```
TEXT$(:3:)
```

So, we changed all our substring references to the proper method. Now our program worked fine. Complete with full screen controls and protected input fields.

We were so happy that our ORDERS could now be processed on our IBM PC compatible (a Zenith 151) that we have ordered a hard disk system for it - to hold all our subscribers.

So, based on the above experience, I am happy to report that COMAL IS COMPATIBLE.

## LOWER CASE PLOTTEXT IN COMAL 0.14

by David Stidolph

One of COMAL 0.14's problems is the inability to put lower case characters on the graphics screen. By the use of one poke, you can change which character set COMAL 0.14 will use for PLOTTEXT. The command for this is:

```
POKE 29260,216
```

One problem, though, is that COMAL stores upper case characters (A-Z) in memory as chr$(193)-chr$(216), instead of chr$(97)-chr$(123), which is the range needed to get the proper character on the graphics screen. A small machine language program converts the upper case characters in the string to the proper values necessary for putting them on the graphics screen. The machine code is completely relocateable, so just change the variable 'start' in CONVERT and CONVERT'SETUP' to the area you where you want the machine code to be.

The procedure PLOT'CHAR will put your string on the graphics screen just as if you were using a 'PRINT AT' command in 2.0 COMAL. Just plug in the ROW and COLUMN numbers for where you want the string, and the string will be put there. PLOT'CHAR(1,1,"Hello Sam") will put the text 'Hello Sam' at the upper left hand corner of the screen. The numbers to use for positioning text are 1-25 for the row, or line number, and 1-40 for the column.

```
//
proc plot'char(x'loc,y'loc,s$)
 convert(s$)
 plottext x'loc,y'loc,s$
endproc plot'char
//
proc convert(ref s$) closed
 start:=2024
 if peek(start)<>160 then
  convert'setup
 endif
 if s$<>"" then sys 2024
endproc convert
//
proc convert'setup closed
 start:=2024
 poke 29260,216
 poke start,160
 poke start+1,3
 poke start+2,177
 poke start+3,51
 poke start+4,170
 poke start+5,200
 poke start+6,177
 poke start+7,51
 poke start+8,201
 poke start+9,192
 poke start+10,144
 poke start+11,4
 poke start+12,233
 poke start+13,96
 poke start+14,145
 poke start+15,51
 poke start+16,202
 poke start+17,208
 poke start+18,242
 poke start+19,96
endproc convert'setup
```

## AUTO RUN YOUR COMAL PROGRAM

You can easily set up a disk so that when you start up COMAL (LOAD "BOOT*",8) that your program will automatically RUN right away. To do this just modify the HI program supplied with most of our COMAL disks so that it includes a CHAIN "NAME" (where NAME is the name of your program). Make sure to save it on disk with the name "HI".

## MINDY MINDY MINDY

Yes, one of the national COMAL advocates is again within our pages. We have permission to reprint 3 of Mindy Skelton's COMAL COLUMNs. So without any further ado, here's Mindy:

## COMAL COLUMN - GRAPHICS

by Mindy Skelton

In an effort to serve the needs and interests of all the COMAL users who read your humble author's monthly endeavors, this month's article will focus on COMAL graphics. (I know that's not what I said last month, but by now you should be used to my little side trips. This one is just longer than usual. We will return to checkbook balancing in subsequent articles.) I presume that many of you were attracted to your Commodore, at least in part, because of its graphic capabilities, only to find that hi-res graphics and sprites were troublesome and frustrating. You are not alone.

One of the many advantages of COMAL is the easy access to graphics which it affords its users. Unfortunately, at this point in time (I've wanted to use that phrase for years!) there are few guides to the COMAL user who wants to tap this graphic bonanza. To temporarily bridge the gap (soon be filled by Len Lindsay's book on COMAL graphics, to be published by Reston Publishing and by a tutorial book by yours truly [both now available and in stock]) we will look this month at some of the more basic graphics commands. Please be aware that this is a VERY BRIEF look. More information will follow in later articles.

Graphic commands can be included in COMAL programs just like any other command, with all the structures to manipulate numbers and words now used to drive your "turtle" or move your sprite. They can also be entered in "immediate" graphic mode, where you give a command and immediately see the result on your screen. In either case, you must activate the graphic mode of your program, by entering the command SETGRAPHIC. The first time you use this command you must also add a type specification, 0 for hi-res and 1 for multicolor. On subsequent activations in the same program or session, the command SETGRAPHIC alone will suffice. To return to the text screen, the command SETTEXT is used. Changing to text screen in no way affects what happens on your graphic screen, so you could have something being drawn on your graphic screen while directions were printed on your text screen and then switch to an already drawn picture. Clever language, this COMAL.

When you use the SETGRAPHIC command you will see your graphic screen with a triangular "turtle" (shades of LOGO) sitting in the middle of the screen (This is home. When you enter the command HOME, the turtle will return here.). The size of the turtle can be changed by the command TURTLESIZE followed by a number from 0 to 10 (default size is 10). If you don't want to see the turtle at all, the command HIDETURTLE will turn her off, and SHOWTURTLE will bring her back.

In immediate mode, there will be a "window" at the top of the screen where your commands will be visible. If you wish to be rid of this window, type FULLSCREEN. To get it back, type SPLITSCREEN or press F3. To get back to your text screen type SETTEXT or press F1.

You can set the color of your screen and border with the commands BACKGROUND (the screen) and BORDER (the edges). Each command is followed by a number from 0 to 15. If you are in hi-res, color changes will not totally take effect until you issue a CLEAR command. The new background color will show around any lines you draw, but the rest of the screen will be unaffected. A CLEAR command clears the screen (except for sprites) and causes the color change to be executed. To change the color of the turtle (and hence your lines) the command you need is PENCOLOR followed by the color you want (indicated by a number from 0 to 15).

Now you're all set. What can you do? You can move that little turtle all over the place! The command FORWARD followed by a specified distance surprisingly moves your little friend forward. BACK plus a distance moves it back the desired distance. (By the way the turtle's "head" is the top of the triangle but she draws from the middle.) You can also move your turtle with the MOVETO, DRAWTO, and SETXY commands. More about them later.

The turtle starts in a vertical position (heading 0). The commands RIGHT and LEFT followed by the number of degrees will turn the turtle's head to its right (clockwise) or left(counterclockwise), the specified number of degrees. Remember the reference is to the turtle's right or left, not necessarily yours. You can also use the SETHEADING command to set the turtle's heading. SETHEADING followed by a degree setting (e.g. SETHEADING 80) positions the turtle at the degree setting requested. This command is very useful to orient yourself before drawing a figure in programs where the turtle is moving around and may have changed its heading.

The turtle normally leaves a line behind it as it moves, but sometimes you may not want it to do this. The commands PENUP and PENDOWN determine whether or not a mark is left.

The hi-res screen gives you a 320*200 screen to work with (that is 0-319 on the X, or horizontal axis, and 0-199 on the Y or vertical axis). The turtle's home at the center of the screen is position 160,99. Many commands take advantage of the coordinate system.

Three commands which use the coordinate system are MOVETO, DRAWTO and SETXY. All of these commands are followed by X and Y coordinates. MOVETO moves you to the desired spot without leaving a line, while DRAWTO and SETXY move you to the specified X and Y location, leaving a line only if the pen is down.

PLOT followed by X and Y coordinates displays that position in the current pen color. PLOTTEXT, followed by X,Y, and a text string, displays the specified text on the graphic screen, with the lower lefthand corner of the first letter starting at the pixel designated by X,Y. Texts can only be displayed in hi-res, and COMAL second-guesses you and "corrects" you if you try to plot one line of text less than 8 pixels up or down from a previously plotted line.

FILL plus coordinates will fill a closed area containing the indicated point. If there is no enclosure, the entire screen will be filled.

The FRAME command defines an area on the screen within which the turtle can leave marks. If the frame is smaller than the screen size, you will still see the turtle when it is outside the frame, but no marks will be left. The four coordinates which follow the FRAME command (x1,x2,y1,y2) are used to determine the lower left corner (x1,y1) and the upper right corner x2,y2) of the frame. The default value of the frame is 0,319,0,199 (the entire screen).

Hope this has cleared up some of the commands for you, and you will go right to your computer to try using them in a program. See you next month!

For those of you who do not already have one, let me recommend the purchase of a tutorial disk. The turtle and graphic tutor programs are excellent (not that the rest of the disk isn't), and will be a real boost to you.

## COMAL COLUMN - MORE GRAPHICS

by Mindy Skelton

First of all, let me apologize for my last column on graphics. I really don't feel it was up to par. To paraphrase Charlie Brown, it read like an article that was written at 4 A.M. the night before it was due, which surprisingly enough, is exactly when it was written. Sigh! Well, onward and upward.

Those of you able to stay awake long enough to finish my last epic, are now familiar with the turtle graphic commands of COMAL. But how, I hear you asking, can I, an ordinary programmer, put these commands together to make a working program? Ask no longer! This very month we will create a visual sampler. Now don't expect an arcade style production here! We are merely going to create a program using the various commands in order to get a grasp of programming syntax and to associate a visual result with the commands. This is not going to be a game you can sell on the open market; in fact the result may strike you as a bit simplistic, but remember, this is a learning exercise. The knowledge you gain here can be transferred to other areas (or to put it more clearly, you can "steal" pieces of this code, or at least its structure to use in more interesting programs).

O.K. Here's our code. Enter this and run it. To stop the demo, just press run-stop.

```
0010 //delete "0:demo.1"
0020 //  mindy skelton
0030 //save "0:demo.3"
0040 //
0050 proc init
0060  setgraphic 0
0070  hideturtle
0080  fullscreen
0090  background 0
0100  border 0
0110 endproc init
0120 //
0130 proc square(h,v,l) (closed)
0140  moveto h,v
0150  for i:=1 to 2 do
0160   forward l-(l/4)
0170   right 90
0180   forward l
0190   right 90
0200  endfor i
0210 endproc square
0220 //
0230 proc poly(h,v,l,ss) (closed)
0240  moveto h,v
0250  for i:=1 to ss do
0260   forward l
0270   right 360/ss
0280  endfor i
0290 endproc poly
0300 //
0310 proc circle(h,v,r) closed
0320  aspect:=1.3
0330  y:=0
0340  first:=true
0350  d'theta:=.1
0360  c:=cos(d'theta)
0370  s:=sin(d'theta)
0380  n:=64
0390  for i:=1 to n do
0400   temp:=r*c-y*s
0410   y:=y*c+r*s
0420   r:=temp
0430   sx:=aspect*r+h
0440   sy:=v-y
0450   if first then
0460    moveto sx,sy
0470    first:=false
0480   else
0490    drawto sx,sy
0500   endif
0510  endfor i
0520 endproc circle
0530 //
0540 init
0550 counter:=0
0560 repeat
0570  counter:=counter+1
0580  scale:=rnd(5,40)
0590  l:=rnd(5,40)
0600  ss:=rnd(3,10)
0610  h:=rnd(0,320)
0620  v:=rnd(0,200)
0630  pencolor rnd(2,15)
0640  ct:=rnd(1,3)
0650  case ct of
0660   when 1
0670    circle(h,v,scale)
0680    if counter>20 then
0690     counter:=0
0700     clear
0710    else
0720     null
0730    endif
0740   when 2
0750    square(h,v,l)
0760   when 3
0770    poly(h,v,l,ss)
0780   otherwise
0790    null
0800  endcase
0810 until key$<>chr$(0)
0820 end
```

This program can be divided into two main sections. Lines 10-520 make up the procedure section, in this case consisting of four procedures: INIT, SQUARE, POLY, and CIRCLE. Lines 540-820 are the main section which calls the procedures, and in this example, sets up randomized values for the variables to be used in our various procedures. Without the main section, the program wouldn't do anything, since a procedure must be called or activated before it performs its function. (That's true of functions too, but the pun involved would just be too awful!)

Let's first look at the procedures of our program:

INIT (lines 50-110)
This procedure will take care of some general housekeeping for us. Line 60 initializes the graphic screen to hi-res by using the parameter 0. Hi-res means we will be drawing lines one pixel wide. The drawings will be delicate, but there will be some problems with colors (More on that later). Line 70 turns off the turtle, which not only makes the drawings neater, but a little faster. Line 80 gives us a full graphic screen by turning off the "window" at the top of the

graphic screen which normally shows the current command. Lines 90 and 100 set the screen and border to black.

SQUARE (lines 130-210)
This procedure will draw squares of varying sizes. Line 90 lists the parameters of the procedure (h,v,l). These parameters receive their values when the procedure is called. H and v will correspond to the x(horizontal) and y(vertical) coordinates of the hi-res screen; refer to last issue if this is unclear. The MOVETO command positions the turtle on these coordinates without leaving a line from the point of origin. If we had wanted to leave a line we could have used DRAWTO. The for-loop in lines 150-200 draws a square by alternately moving the turtle for the distance specified by variable "l" (length), and turning the turtle 90 degrees to the right. You may be wondering why, since we are drawing a square, we don't just go forward and right four times, instead of fudging the value of l as we do in line 160. Well, pixels being the shape they are (rectangular), we need this fudge factor to prevent drawing oblongs rather than squares. (If you think we have problems here, just look at Proc CIRCLE!) Be sure to note that a COMAL for-loop ends with an ENDFOR rather than a NEXT. COMAL is forgiving, however. If you forget, it will put the right syntax for you. What a nice language!

POLY (lines 230-290): This procedure draws multi-sided figures of varying sizes. The number of sides (ss) is randomly assigned in the main program section (later). The counter of our loop is set to the number of sides, and for that many times, the turtle (now invisible) moves forward l (length) and right 360/ss, giving some figure from a triangle to a decagon.

CIRCLE (lines 310-520)
In the discussion of procedure SQUARE, I mentioned the distortion problem caused by the shape of the pixels. This distortion also causes circles to become ovals. Procedure CIRCLE corrects this problem by using an adaptation of a procedure by Kevin Quiggle. Using the coordinates designated by h and v as the midpoint of a circle, and r as the radius of a circle, this procedure plots points around the mid-point and produces a circle. This is one of those times when it is perfectly alright to use a procedure without knowing quite why it does what it does. If you really need to know how the points are determined and plotted, write me and I'll send you an explanation.

Main Section (lines 540-820):
Line 540 calls the procedure INIT. Line 550 initializes a counter to zero. Lines 560-810 set up a repeat loop which will call the graphic procedures and will continue until you press any key. The reserved variable KEY$ [pp. 170-171 in The COMAL Handbook, edition 2] checks to see if any key has been pressed. As long as no key is pressed, the value returned will be CHR$(0). Line 570 increments the counter and lines 580-640 assign random values to the variables which will be used. The random number assignment in COMAL is RND(lower limit, upperlimit). It's so easy! The number generated will be somewhere between the two given numbers, inclusive. The variable ct is used to randomly determine which figure will be drawn. The CASE structure (which we've looked at before) calls a different function for each value of ct. Lines 680-730 check to see if the upper limit of our counter (arbitrarily set to 20) has been reached. If it has, the screen is cleared, the counter is reset, and the program starts to fill the screen again.

If everything went well when you ran your program, you should have filled your screen with shapes of different sizes and colors. Some fun huh? Nice clean lines weren't they? A little messy when they crossed each other though. We'll fix that in a minute.

Now that you've run your demo a few times, let's try something a little different. Add the following code, just after line 670 and run your program again.

```
0671    if counter mod 2=0 then
0672     fill (h),(v)
0673    else
0674     null
0675    endif
```

Well, that gave us a little more variety, but really a mess when they cross each other. That's because we're in hi-res and in hi-res, within any given 8x8 pixel

block each pixel can either be on or off. This, in effect, means you can only have one color and the background. Hum! In order to get around that, we'll have to switch into multicolor mode. In multicolor mode, instead of dealing with an 8x8 block we are dealing with a 4x8 block, but each block is made up of 4 pixels. This means that your lines will be two pixels wide, and therefore less precise, but each block can be one of four colors (two character colors and two background colors), rather than the two color choices in hi-res.

To initialize your screen to multi-color mode, change line 60 to read SETGRAPHIC 1. Now run the program again and note the differences.

I hope this has helped clear up a few things for you, and that now you'll try some programs on your own.

## COMAL & BASIC - A COMPARISON

by Mindy Skelton

Let me first acknowledge that I am stealing the idea for my topic this month from Colin Thompson, our California COMAL connection (of course, with Colin's permission). Just as an aside I hope you all read Colin's article on COMAL in RUN Magazine. I also hope you noticed the word COMAL prominently displayed in the center of the cover. COMAL is finally getting national attention (yea!). Back to the issue at hand - a comparison of COMAL and BASIC.

I know it's confusing for some of you to try and shift from programming in BASIC to programming in COMAL. There's so much new to learn (CASE, and WHILE-DO, and REPEAT-UNTIL, etc., etc.), and there's a whole new way of thinking about programs. It takes a while to get used to structured programming, especially using procedures and functions. We're not used to breaking programs down into smaller and smaller pieces, we're used to building nest of GOTOs and GOSUBs. Have you ever tried to work your way through someone else's program (surely never one of your own!) and gotten lost? It may look like a COMAL program is getting out of hand with procedure after procedure, but it's much easier to find you way around. Sigh, I promised myself I wouldn't get into proselytizing this month.

I've been hammering away at you for several months now about how wonderful COMAL is, and what wonderful things it can do. That's all true, but maybe it would be interesting this month to take a different tack and look at how BASIC (with which you are mostly all familiar) and COMAL would do the same things.

For example, to change the screen color in BASIC, you enter the command:

```
POKE 53281,<color#>
```

Well, you could do that in COMAL too,or you could say:

```
BACKGROUND <color#>
```

Which of those methods seems easier to you?

Let's look at a couple of other fairly common things we do in programs or immediate mode:

| COMAL | BASIC |
|---|---|
| BORDER <color#> | POKE53280,<color#> |
| PENCOLOR<color#> | PRINT"[CTRL + #]"<br>or PRINT"[C= key +#]" |
| SIZE | PRINT FRE(0)-<br>(FRE(0)<0)*65536 |
| CHAIN <program name> | LOAD <program name><br>RUN |
| CAT<br>(non-destructive) | LOAD"$"<br>(destructive) |
| DELETE "0:file" | open 1,8,15:print#1,<br>"s0:file":close 15 |
| DIM chart (4,4)<br>OPEN 2,"file",READ<br>READ FILE 2:chart<br>CLOSE 2 | DIM chart (4,4)<br>open 2,"file,s,r"<br>for i=1 to 4<br>for j=1 to 4<br>input#2,chart(i,j)<br>next j<br>next i<br>close 2 |

Let's look at some code examples you might use. How about a loop to slow down a program.

BASIC

```
10 for i = 1 to 500
20 next
30 rem  causes program to pause
```

COMAL

```
10 for wait:=1 to 500 do null
20 // causes program to pause
```

Or how about waiting for a key to be pressed?

BASIC

```
10 get x$:if x$="" goto 10
```

COMAL

```
10 while key$=chr$(0) do null
```

Maybe you want to wait for a particular key, for example the RETURN key.

BASIC

```
10 get x$:if x$<>chr$(13) goto 10
```

COMAL

```
10 while key$<>chr$(13) do null
```

Let's try something a little more complicated. Let's look at some code that waits for one of a limited number of acceptable responses, evaluates the response, and then, based on the evaluation, branches to other possible actions. A piece of code like this could be used as the core of a menu-driven program.

BASIC

```
10 input "press F1, F2 or F3";r$
20 get r$:if r$="" goto 20
30 if r$=chr$(133) goto 100: REM if F1
     pressed then goto another routine
40 if r$=chr$(137) goto 200: REM if F2
     pressed then goto another routine
50 if r$=chr$(134) goto 300: REM if F3
     pressed then goto another routine
60 print"not a valid response."
70 goto 20
```

COMAL

```
 10 dim reply$ of 1
 20 print"press F1, F2 or F3"
 30 repeat
 40  reply$:=key$
 50  case reply$ of
 60  when chr$(133) // when F1 pressed
 70   task1 // calls procedure task1
 80  when chr$(137) // when F2 pressed
 90   task2 // calls procedure task2
100  when chr$(134) // when F3 pressed
110   task3 // calls procedure task3
120  otherwise
130   if reply$<>chr$(0) then print "No
      t a valid response." //wrap line
140  endcase
150 until reply$=chr$(133) or reply$=ch
    r$(137) or reply$=chr$(134)//wrapln
```

I hope these side-by-side comparisons have been helpful to you. Be sure to get one or more of the new COMAL demo disks available from COMAL Users Group U.S.A, Ltd. Try writing some COMAL programs this month incorporating pieces of this code, or modifying or improveing on a demo, or even something entirely your own. Let's get programming out there. Have fun.

# FIND COMMAND FOR COMAL 0.14

by Doug Weick

FIND for COMAL 0.14 is driven by multiple interrupts to allow for immediate mode interaction. It is necessary that the cursor be in the home position before type in the word, variable, function, procedure, or any other string you wish to search for. Only the first 10 string characters are significant! Additional characters are ignored by the interpreter. Always terminate your search string with a back arrow (the <- key in top left corner of keyboard) which is used as a string delimiter as well as an interrupt branch. Entering a british pound sign (near top right side of keyboard) followed by a back arrow (<-) can be used to kill FIND 0.14 and reset the default interrupt. A back arrow (<-) can still be used during programming without interfering with FIND, as long as it isn't typed on the top screen line.

After you type the back arrow, the cursor will jump to the bottom of the screen and a flashing "E" will appear. You now press the RETURN key and see the lines in which your search string occurs indicated by a white back arrow. Use CONTROL key to slow listing as desired. And STOP key to stop the listing. You may also type EDIT and a line range over the flashing E before pressing RETURN in order to target your search to a specific program area. A search may be performed as many times as necessary without retyping the search string. Just position the cursor on the bottom line and type EDIT <range> and RETURN.

A disk directory can be searched in the same manner by typing "CAT"<space> then RETURN. To clear the current search string from memory when your search is completed, type CONTROL C. A new search string cannot be entered until the current one has been designed so that it will not crash your program even if you make a mistake. If, however, you plan to use interrupts with your program, first disable FIND by entering a british pound sign and back arrow at the cursor home position on the screen. FIND can be restarted with SYS 53077.

FIND can be modified to suit your own preferences by using the following chart:

```
PROGRAM AREA: $CDAA-$CF7A
              52650-53114

ENTRY POINT:  $CF55
 (Start = SYS 53077)

DEFAULT PENCOLOR IN $CDCB / 52683
 (default is 14, lt blue)

DEFAULT LINE INDICATOR IN $CE04 / 52740
 (default is 31 - screen code for <-)

DEFAULT INDICATOR COLOR IN $CDFC / 52732
 (default is 1, white)

DEFAULT CONTROL CHAR IN $CF48 / 53064
 (default is 20, keyboard code for C)

SEARCH MODE SETTINGS:
 192 = EDIT MODE VALUE (DEFAULT)
 197 = LIST MODE VALUE

MODE STORAGE LOCATIONS:
$CDD3   $CE0E   $CF4C
52691   52750   53068

SCREEN DISPLAY LOCATIONS:
[Flashing E response]
$CE59   $CE5E   $CE63   $CE68
 (E)     (D)     (I)     (T)
```

The value in the MODE STORAGE LOCATIONS controls the position at which the string search is started on the screen. In the EDIT search mode each line is searched starting with the first character on the screen. This includes line numbers. In the LIST search mode each line is searched starting with the 6th screen character. Line numbers are not detected in this mode. The program default is to the EDIT search mode.

Target strings broken by a screen line will not be detected. Any of the default values can be changed by poking desired values into the corresponding location or inserting them into the data statements in this program. Each DATA line number corresponds to the last 4 digits of its address in memory: 5 location values per line. For example, if you wanted to change the default line indicator character (<-) to an asterisk, just change the value 31 to 42 in line 2740.

A target string may not be detected if it occurs in the last 2 program lines, since they are pushed up to make room for the cursor faster than the interrupt can

process them. This problem can be overcome by using the CONTROL key to slow down the listing. Target strings that occur consecutively in blocks (such as DATA statements), may be indicated as they occur intermittantly instead of on every line. Because of screen code differences, FIND will not detect UPPER case characters in the lower case mode but will detect all characters in the upper case mode.
This program is on TODAY DISK #6.

## SETPAGE

SETPAGE is a 2.0 COMAL command in the 'system' package that gives access to any part of memory in the Commodore 64. Since the 64 can only address 64k of memory, a technique called overlays is used. Overlays mean that multiple memorys are 'stacked' on top of each other, and the areas that are needed can be 'flipped' into the addressing space of the 64.

The following is a list of the numbers to use to get at the different areas of memory.

```
Num Area
--- ----
  0 All Ram (64k)
  1 BASIC
  2 Kernal
  3 BASIC and Kernal
  8 I/O and Color memory
$80 COMAL overlay #0 (Power up routines
      math routines, and packages:
      DANSK, ENGLISH, and SYSTEM)
$81 COMAL overlay #1 (editor, syntax
      analysis and code generation,
      prepass (SCAN), recreator (LIST)
$82 COMAL overlay #2 (Runtime-module)
$83 COMAL overlay #3 (other packages)
$84 COMAL overlay #4 (EPROM expansion)
$85 COMAL overlay #5 (EPROM expansion)
```

To use this table, just put in the number for the area you want to look at and put it in the SETPAGE command. For example, if you wanted to look at the RAM under the Kernal (the graphics bitmap) the command SETPAGE(0) would be used. If you wanted to look at the GRAPHICS package (maybe to disassemble the code) you would use the command SETPAGE($83).

Remeber, SETPAGE is a command in the SYSTEM package, so the command 'USE system' must be used before the command SETPAGE is used.

## DEFINE FUNCTION KEYS BATCH FILES

by David Stidolph

Batch files are disk files containing commands that can be executed just as if you typed them in from the keyboard. This powerful feature (usually found only on main frame computers) can save you many hours of typing by placing often used commands together in one file where they can be executed with the SELECT INPUT command. The following examples are a set of commands used to define the function keys. They could be on disk and setup with one command:
SELECT INPUT "filename"

```
USE system
defkey(1,"RENUM"13"")
defkey(2,"MOUNT"13"")
defkey(3,"USE turtle"13"")
defkey(4,"AUTO ")
defkey(5,"EDIT ")
defkey(6,"LIST ")
defkey(7,"RUN  "13""11""13"")
defkey(8,"SCAN"13"")
```

This is the default setting of the function keys. The "13"" you see in some the the commands above means you want a carriage return, CHR$(13), put in. The F7 key is defined above so that you can put the cursor on a line from a CATALOG of a disk, and run that program. The text "RUN " will type over the number of blocks the file takes up, and type a carriage return. An error will now occur because of the file type PRG after the file name. COMAL will automatically place the cursor on the PRG and give you an error message. The rest of the key definition comes into play now. The "11"" will erase the PRG (CHR$(11) means "erase the rest of the line"), and the second carriage return will enter the now correct line. The program will now be loaded and run.

```
USE system
defkey(1,"MOUNT "13"")
defkey(2,"MOUNT ""1:""13"")
defkey(3,"CAT ""0:""13"")
defkey(4,"CAT ""1:""13"")
defkey(5,"UNIT$""0:""13"")
defkey(6,"UNIT$""1:""13"")
defkey(7,"RUN  "13""11""13"")
defkey(8,"COPY "13",""1:*"13"")
```

This function key definition is for owners of dual disk drives. Function keys F1, F3 and F5 are for drive 0 and

function keys F2, F4 and F6 are for drive 1. MOUNT means to initialize (not format) a disk, CAT will give you a directory of a disk, and UNIT$ changes the default drive. The last definition, COPY, can copy a file from drive 0 to drive 1. Just put the cursor on the same line as the directory listing of the file you want to copy, press the f8 key, and the file will be copied.

```
USE system
defkey(1,"MOUNT"13"")
defkey(2,"CHANGE""settext"", ""textscre
en"""13"") // wrap line
defkey(3,"ENTER "13""11""13"")
defkey(4,"CHANGE""clear"", ""clearscree
n"""13"") // wrap line
defkey(5,"("13")"13"")
defkey(6,"FIND ""identify"""13"")
defkey(7,"("13""157")"13"")
defkey(8,";showsprite(")
```

This set of function keys is for translating a COMAL 0.14 program which have been listed to disk, to a COMAL 2.0 program. When you ENTER a program (F3) and COMAL 2.0 finds a problem, like no parentheses around the graphics statements, you can use the F5 and F7 keys to add them. You use the F7 key when the line has a comment, otherwise the right hand parenthese would go AFTER the comment, not before it. After the program has been successfully ENTERed into memory, press the F2 key, and press the RETURN key when each line appears. Do the same with the F4 key. Now all refereces to "clear" and "settext" have been changed to the right values. Since INDENTIFY will no longer turn on the sprite, use the F6 key to locate the INDENTIFY statements and the f8 key to add SHOWSPRITE to the line (you have to fix the rest of the line yourself).

```
USE system
defkey(1,"LIST ")
defkey(2,"RENUM "13"")
defkey(3,"SAVE  "13""11""13"")
defkey(4,"DELETE "13""11""13"")
defkey(5,"FIND "" PROC """13"")
defkey(6,"FIND "" FUNC """13"")
defkey(7,"RUN "13""11""13"")
defkey(8,"TRACE"13"")
```

This last function key list is for general work on programs. The F5 and F6 keys will list all procedures and functions in your program. TRACE is used when your program stops because of an error. It is particularly usefull when your program stops inside of a procedure.

## SPACE REQUIRED AROUND COMAL WORDS

When entering a COMAL program remember to include a space after each COMAL keyword. For example, if AGE is a variable, any you wish to print it, the following would be INCORRECT:

```
PRINTAGE
```

Many BASIC programmers have a bad habit of SQUISHING their program lines since the program in BASIC will run faster. In COMAL, the above example is really calling a procedure called PRINTAGE. We must include at least once space after the word PRINT to have a correct COMAL line. Note: you may include more than one space - COMAL just ignores the extras. In COMAL the line would look like:

```
PRINT AGE
```

## LIBRARY DISK/BOOK ERRATA

This is a complete errata sheet (as of Jan. 4, 1985) for Kevin Quiggle's Library of Procedures and Functions. Dir.l was missing from the original disk, but is included on TODAY DISK #6.

The error in DAY'OF'WEEK seems to rest in the INT function. Kevin told us that in 0.14 the command PRINT INT(.6*15) evaluates as 8, while in 2.0 the same command evaluates to 9. If this difference stems from COMAL, there may be some problems in the future in translating between the two versions.

PROCEDURE KOALA:

In line 9010, change "port" to "p"

FUNCTION DAY'OF'WEEK:

In line 900, change ".6" to ".61"

PAGE 23 in book:

Change "wait" to "inkey" (although inkey is not in proper alphabetical order, it is properly indexed and otherwise ok).

# WHERE HAVE ALL THE GOTO'S GONE

by Raymond Quiring

GOTO has received a bad name ever since BASIC was invented and teachers realized the mistake. (That's right. GOTO and BASIC are both held in low esteem.) Structured languages have learned to do without GOTO and it has served as a convenient whipping boy for those who dislike any kind of programming slopiness. But, what is so bad about GOTO? Is it really a threat to society? Would we all sleep better knowing that our programs are free of GOTO's? Only if we deceive ourselves about the missing GOTO's.

You see, it is impossible to do without GOTO'S. The C64 couldn't exist without a plethora of GOTO's. PASCAL and COMAL are full of them. Every programming language with structure contains GOTO's in abundance. But, they are hidden from our eyes and thoughts, cleverly disguised within such words as ENDIF, ENDWHILE or ENDFOR. Even honest BASIC with its flaunted GOTO hides it in NEXT, RETURN and, not so cleverly, in GOSUB. The GOTO's are hidden for a reason not too dificult to understand.

We all know how a FOR NEXT loop works. We know it so well that we may have forgotten the stupid questions we had to ask about what NEXT was supposed to do.

```
100 FOR I=1 TO 55 STEP 3
110 PRINT "GOTO";I
120 NEXT I
130 ...
```

NEXT is a composite command: Add 3 to I and test I against 55. If I<= 55 then (you guessed it) GOTO 110. If I>55 then GOTO 130. Simple, easy to understand. Here is NEXT as it would appear in a primitive language:

```
NEXT: 120.1 I=I+3
      120.2 IF I>55 THEN GOTO 130
      120.3 IF I<=55 THEN GOTO 110
```

The invention of the word NEXT to replace three lines of code has far reaching implications. The mind deals with NEXT more easily than the code it replaces. The mind conjures up, seemingly without effort, the entire idea of NEXT. We deal with it on a higher plane of thought and our programming becomes more efficient and interesting. We are able to work with increasingly more complex ideas because they are condensed into a single word. The effect is to replace a primitive idea like "one with wrench who makes water flow in pipes" with a simple word "plumber". This is the real reason we should avoid GOTO: it enslaves our mind to a guttural time consuming discourse with our computers.

Comal contains many structures that may seem strange and forbiding to the BASIC programmer:

```
REPEAT ... UNTIL
IF ... THEN ... ELSE ... ENDIF
WHILE .. DO .. ENDWHILE
CASE ... OF ... WHEN ... OTHERWISE ...
ENDCASE
PROC ... ENDPROC
```

These structures condense tons of programming into a few simple, easily understood ideas. They all serve to make programming more efficient and enjoyable.

GOTO's lurk in far more interesting places than the structures of COMAL. Look again at the lines of code at 120.1 which represent NEXT. The designer of BASIC has given us a word that is useful and efficient. Wouldn't it be nice if we could make our own words and use them just as we use NEXT? The FUNCTION command in BASIC is something like that. But it is so primitive and limited that it is almost not worth mentioning. The closest idea to this in BASIC is the subroutine. You "call" (there's a nice idea condensed into a single word) a subroutine by saying GOSUB 1030. They didn't put much effort into hiding the GOTO!

GOSUB: before you GOTO 1030 save the next line number so you know where to RETURN.

RETURN: after you finish the subroutine at 1030 GOTO the next line following the GOSUB.

These are simple ideas compressed into a word which helps make programming a pleasure. But, how much more agreeable to the mind to "call" a subroutine by its human name rather than a number.

The designer of COMAL has given us the ability to make subroutines and give them

names that are used exactly like any other built in word. We can extend the language to include more complex ideas that are "called" merely by the use of the new word. These are the PROC's and FUNC's of COMAL.

This is the real power of COMAL. Design a PROC (a word) to print a file and call it PRINTFILE(filname$). Then whenever you want to print "myfile'name" merely write PRINTFILE(myfile'name) in the program as if it were a KEYWORD. All the hidden GOTO's get busy and direct the computer to perform the procedure you defined in PRINTFILE.

The various COMAL structures may seem weird and confusing at first. Just remember that buried in those UNTIL's and ENDPROC's are little programs where GOTO is king. Leave the GOTO's there. Let your computer follow them around; it just loves the exercise. Learn the the new ideas and you will be rewarded with urbane programming conversation far removed from the gutteral GOTO.

## EPSON FX-80 PRINTER CODES

by Mark Finley

You can use these procedures to control the Epson FX-80 printer. They are similar to the standard "EPSON" command set, so you could change them to suit.

```
//delete "proc.fx-80 cmds"
//list   "proc.fx-80 cmds"
//by Mark H. Finley
//
PROC beep
 PRINT CHR$(7),
ENDPROC beep
//
PROC backspace
 PRINT CHR$(8),
ENDPROC backspace
//
PROC enlarge
 PRINT CHR$(27),CHR$(14),
ENDPROC enlarge
//
PROC unenlarge
 PRINT CHR$(27),CHR$(20),
ENDPROC unenlarge
//
PROC condense
 PRINT CHR$(27),CHR$(15),
ENDPROC condense
//
PROC uncondense
 PRINT CHR$(27),CHR$(18),
ENDPROC uncondense
//
PROC underline
 PRINT CHR$(27),"-",CHR$(1),
ENDPROC underline
//
PROC nonunderline
 PRINT CHR$(27),"-",CHR$(0),
ENDPROC nonunderline
//
PROC emphasize
 PRINT CHR$(27),"E",
ENDPROC emphasize
//
PROC unemphasize
 PRINT CHR$(27),"F",
ENDPROC unemphasize
//
PROC bold
 PRINT CHR$(27),"G",
ENDPROC bold
//
PROC unbold
 PRINT CHR$(27),"H",
ENDPROC unbold
//
PROC elite
 PRINT CHR$(27),"M",
ENDPROC elite
//
PROC pica
 PRINT CHR$(27),"P",
ENDPROC pica
//
PROC superscript
 PRINT CHR$(27),"S",CHR$(0),
ENDPROC superscript
//
PROC endscript
 PRINT CHR$(27),"T",
ENDPROC endscript
//
PROC subscript
 PRINT CHR$(27),"S",CHR$(1),
ENDPROC subscript
//
PROC proportional
 PRINT CHR$(27),"p",CHR$(1),
ENDPROC proportional
//
PROC unproportional
 PRINT CHR$(27),"p",CHR$(0),
ENDPROC unproportionl
//
```

## COLOR PICTURES FOR COMAL 0.14

by David Stidolph

COMAL 0.14 has published programs for loading and saving black and white pictures, but they are not compatible with the new 2.0 cartridge. I wrote a machine language program to emulate the LOADSCREEN and SAVESCREEN commands of 2.0 so that COMAL 0.14 users would be able to share pictures.

To LOAD or SAVE color pictures from your own program is easy. All you have to do is have this machine language program on disk, and add three small procedures to your program. One procedure is for loading the machine language program into memory, and the other two are for loading and saving the color picture. Hi Res pictures are saved as 36 block sequential files, and multi-color pictures are saved as 40 block sequential files. When a picture is loaded, the border and background color is changed, even if the textscreen is on instead of the graphics screen.

The procedures are show below with a sample program. The machine code program is on COMAL Today Disk #6 because it is too long to list in this newsletter. A small bug exists that interferes with the use of KEY$. If you wish to use KEY$ after you load or save a picture include the command: POKE 198,0 first (this also clears the current keyboard buffer).

```
// delete "0:loadscreen'demo1"
//  by David Stidolph
// save   "0:loadscreen'demo3"
// load machine code routine
load'obj("load/save.mem")
// initialize graphics screen
setgraphic 0
// load graphics picture
loadscreen("hrg.world'map")
// clear keyboard buffer
poke 198,0 // needed!
// wait for key to be pressed
while key$=chr$(0) do null
//
proc loadscreen(name$)
 open file 2,name$,read
 poke 189,2 // store file number here
 sys 52398
endproc loadscreen
//
proc savescreen(name$)
 open file 2,name$,write
 poke 189,2 // store file number here
 sys 52224
endproc savescreen
//
proc load'obj(name$) closed
 poke 858,169
 poke 859,0
 poke 860,76
 poke 861,213
 poke 862,255
 open file 78,name$+",p",read
 sys 858
 close file 78
endproc load'obj
```

## CAUGHT IN A LOOP - A COMMON THING

Remember - you have to return sometime! If not, you will be caught in an infinite loop, a programmers nightmare. Programs that use a MENU often unwittingly do this if they have each choice end by once again calling the MENU (a no-no). Bill Brogie of Pomona California had this problem. He wrote a very nice program called EASY TURTLE for use by grade school students. It would always 'crash' after only 101 Easy Turtle commands were given. For now, they use the error (STACK OVERFLOW) to signal that the students turn is up. But the solution to his program's problem was simple. Here is what we advised him:

The problem with your program is an endless loop. You have a procedure GET'COMMAND which calls DO'COMMAND. Then the problem is that at the end of DO'COMMAND you call GET'COMMAND. The fix is easy. You would like to continually call the GET'COMMAND procedure. Now, remember, when a procedure ends, program execution continues from the point that called it. So, just let DO'COMMAND end. It will return to GET'COMMAND. Let GET'COMMAND end as well. Then it will always return to the original GET'COMMAND call. Now, simply put that call inside a WHILE or REPEAT loop like this:

```
WHILE EASY'TURTLE DO
  GET'COMMAND
ENDWHILE
```

The program now works fine.

## COMAL 2.0 MODEM PROCS & FUNCS

The following are some procedures and functions that you can use in COMAL 2.0 programs to control your Commodore 1650 or Westridge modems.

```
PROC call(number$)
  TIME 0
  hang'up //   reset phone line
  pause(2) //  allow 2 seconds
  off'hook //  allow one second
  pause(1) //  for dial tone
  dial(number$) // do it
  waitcarr(15) // 15 seconds or carrier
ENDPROC call
//
FUNC carrier
  RETURN NOT PEEK(56577) BITAND 16
ENDFUNC carrier
//
PROC waitcarr(sec)
  TIME 0
  REPEAT
  UNTIL carrier OR TIME>=(sec)*60
ENDPROC waitcarr
//
PROC hang'up
  POKE 56579,32
  POKE 56577,0
ENDPROC hang'up
//
PROC off'hook
  POKE 56579,32
  POKE 56577,32
ENDPROC off'hook
//
PROC dial(number$)
  FOR i:=1 TO LEN(number$) DO
    PRINT number$(i),
    IF number$(i) IN "0123456789" THEN
      n:=VAL(number$(i))
      IF n=0 THEN n:=10
      FOR pulse:=1 TO n DO
        POKE 56577,0
        pause(.05)
        POKE 56577,32
        pause(.05)
      ENDFOR pulse
    ELIF number$(i)="," THEN
      pause(1)
    ENDIF
    pause(.4)
  ENDFOR
ENDPROC dial
//
PROC pause(delay)
  TIME 0
  WHILE delay*60>TIME DO
  ENDWHILE
ENDPROC pause
//
FUNC modem'get$(f'num) CLOSED
  old'lo:=PEEK($0324)
  old'hi:=PEEK($0325)
  DIM c$ OF 1
  TRAP
    TRAP ESC-
    POKE $0324,PEEK($032a)
    POKE $0325,PEEK($032b)
    c$:=GET$(f'num,1)
    POKE $0324,old'lo
    POKE $0325,old'hi
    TRAP ESC+
    RETURN c$
  HANDLER
    POKE $0324,old'lo
    POKE $0325,old'hi
    TRAP ESC+
    REPORT // pass error outside
  ENDTRAP // of this routine.
ENDFUNC modem'get$
```

## SONG EDITOR FOR COMAL 2.0

by Mark Clifford

The SONG EDITOR is used to prepare and edit songs for the cartridge playscore command. It begins by asking if you wish to enter an new song or edit one previously saved. If you choose a new song you will be asked to enter the maximum number of notes for each voice so that the arrays can be DIM'ed. If you chose an old song, the name is requested and the song file is loaded.

The main menu appears next. The options are to enter or change the asdr and wave form; enter the notes for a voice; play the song; save the song or quit.

The first option asks for the attack, decay, sustain, release, wave form, and, if a pulse wave is selected, the pulse width. These values are saved in a 3X6 array along with the note and time values. These values will be loaded with the song, but need not be used.

The second option first asks for the voice you wish to enter, and if any values are already present, the note number you wish to start with. Notes are entered in the format used by the NOTE command (c4, c4#,etc) and converted to frequency values. The current frequency value is converted to the NOTE format and displayed on the input line so that it can be reentered without change by pressing <return>. The time values to be used are displayed on the screen. All time entries are multiplied by 4, the value of the variable tempo, so that most time values can be entered as one digit numbers. A rest can be included in the previous note's time off or entered as R0. A block of notes can be repeated by entering RP for a note value; you will then be asked for the numbers of the notes which begin and end the block to be repeated. Entry of RE stops the editing without altering the current note, and returns to the main menu.

The play option allows you to stop any of the voices so you can hear only one or two voices if you wish.

The save option prompts with '0:sng.' for song name and requires either a 0 or a 1 as the first character. If you enter a name already on the directory you are asked if you wish to replace the current file.

The program is modeled on my BASIC editor published in the October issue of RUN as Sid Seranade.

The SONG PLAYER contains the procedures needed to load and play one of the song files.

The files beginning with 'sng' are song files. The two polkas are the same except for the asdr; one sounds like an accordian, the other like a banjo.

This program is on TODAY DISK #6.



**COMAL FROM A TO Z / TPUG COMAL GUIDE**

Both COMAL FROM A TO Z from COMAL Users Group USA and COMAL REFERENCE GUIDE from TPUG are based on the same manuscript from Borge Christensen. If you have one, you do not need the other.

## 2.0 PACKAGES EXPAND COMAL

by David Stidolph

The creators of 2.0 COMAL wanted to make programs truly portable between various machines running 2.0 COMAL, so they defined the main language without machine dependent features (no graphics, sound, sprites, etc.). All machine dependent features were put in separate PACKAGES which add the commands to the cartridge, without putting them into the main definition.

A package is a group of machine code routines which can be called by name and can pass variables or other information, like any regular procedure or function. Until a package is activated with the USE command, the commands in the package are not known to the system. To activate a package the command 'USE name' is used, and the system will add the command names to the name table so they can be used. An example is:

```
USE graphics
```

The advantage of packages is twofold. Commands in packages can be redefined (maybe for a different printer, etc.) and you can add your own packages to your program. Jesse Knight has completed a book / disk set COMAL 2.0 PACKAGES explaining how to create a package. A package can be added to your program using the LINK command.

The LINK command goes out to the disk drive and reads in a file (package) produced from Cmmodore's Assembler. Now the machine code program is 'linked' to your program, and when you save the program, the package is saved with it. This is true of all regular packages, but there are some package types that are not saved with the program. One example of this is the French error message package on demo disk #1 ("pkg.francais"). The package itself will tell what type it is. To LINK in the French error messages, the following command is used (with demo disk#1 in the disk drive):

```
LINK "pkg.francais"
```

Remember, the LINK command only loads the package. The USE command is needed to add the commands into the name table so you can use them. With the French error messages loaded into memory, the command "Use francais" will change the error messages to the French error messages in the package.

Packages can be called when the cartridge is first turned on, right before the COMAL editor is activated, when the RUN/STOP and RESTORE keys are pressed, when the program stops for an error, or when the following commands are used: LINK, DISCARD, NEW, CHAIN, RUN, CON, BASIC

As you can see, a package can interface with COMAL at any level, and all the commands and routines of the cartridge, BASIC, or the kernal are accessable to the assembly code programer (80K of ROM). COMAL is the first micro language to give such easy access to machine code, and to give machine code easy access to variables and output back to the calling program.

## EXPAND RAM PROGRAM CORRECTION

On page 19, COMAL TODAY #5, we left out one IMPORTANT line of the EXPAND COMAL program. The program is listed below correctly, the added line in bold.

```
// delete "0:expand'memory"
//  by john mccoy // corrected !
// save   "0:expand'memory"
print chr$(147),
expand'ram
print "type 'new' and 'size' to"
print "see expanded memory"
end
//
proc expand'ram
 b391:=hex'dec("b391")
 b39d:=hex'dec("b39d")
 for addr:=b391 to b39d do
  poke addr+1024,peek(addr)
 endfor addr
 poke hex'dec("11d9"),183
 poke hex'dec("3b98"),183
 poke hex'dec("4432"),183
 poke hex'dec("4928"),183
 poke hex'dec("5549"),183
 poke hex'dec("5f0c"),183
 poke hex'dec("601e"),183
 poke hex'dec("68f2"),183
 poke hex'dec("7027"),183
 poke hex'dec("7037"),183
 poke hex'dec("7519"),183
 poke hex'dec("8689"),183
 poke 2066,144
 poke 2067,183
endproc expand'ram
//
func hex'dec(h$) closed
 l:=len(h$)
 if l=0 then
  return 0
 else
  d:=ord(h$(1:1))-48
  if d>9 then d:=d-7
  for i:=2 to l do
   a:=ord(h$(i:i))-48
   if a>9 then a:=a-7
   d:=16*d+a
  endfor i
  return d
 endif
endfunc hex'dec
```

## EPYX FAST LOAD CART WORKS

We received a note that the EPYX FAST LOAD Cartridge works and will LOAD COMAL 0.14 in 30 seconds.

## CALL IN TECHNICAL COMAL QUESTIONS

If you are a COMAL TODAY subscriber you can call us between 11 am and 3 pm every Monday, Wednesday, and Friday and someone should be available to answer your questions (please be brief - there are alot of others waiting to call in too). This is available only to subscribers. Or, you also can send a Self Addressed STAMPED envelope with your questions and we will send you a reply.

INFO PHONE NUMBER: (608) 222-4432

## FILE NAME CONVENTIONS FOR COMAL

Files can be used for many things. By following a simple naming convention, you can make your file names meaningful to yourself as well as others. In COMAL 0.14 we added on an "extension". In COMAL 2.0 we add a "prefix". This allows you to take advantage of the built in pattern matching of DIR or CAT. To see all the Sprite Shape files on the disk you would just issue the command: CAT "SHAP.*"

COMAL 0.14 NAME CONVENTIONS

| | |
|---|---|
| NAME | COMAL program |
| NAME.L | Program LISTed to disk |
| NAME.PROC | Procedure LISTed to disk |
| NAME.FUNC | Function LISTed to disk |
| NAME.DAT | Data file |
| NAME.TXT | Text file |
| NAME.EZ | EasyScript file |
| NAME.PCLP | PaperClip file |
| NAME.WPRO | WordPro file |
| NAME.HRG | Hi Res Graphics Picture file |

COMAL 2.0 NAME CONVENTIONS

| | |
|---|---|
| NAME | COMAL program |
| LST.NAME | Program LISTed to disk |
| DSP.NAME | Program DISPLAYed to disk |
| PROC.NAME | Procedure LISTed to disk |
| FUNC.NAME | Function LISTed to disk |
| EXT.NAME | External procedure/function |
| SHAP.NAME | Shape file for sprites |
| FONT.NAME | Font file |
| DAT.NAME | Data file |
| TXT.NAME | Text file |
| EZ.NAME | EasyScript file |
| PCLP.NAME | PaperClip file |
| WPRO.NAME | WordPro file |
| PKG.NAME | Package file |
| SNG.NAME | Song file |
| HRG.NAME | Hi Res Graphics Picture file |
| BAT.NAME | Batch file |

## RESTORE TO LABEL IN COMAL 0.14

by John Eldredge

This procedure emulates the RESTORE <label> command of COMAL 2.0. To use this procedure, EXEC RESTORE'LABEL(STRING$), where STRING$ is either a variable or a literal string in quotes. The data pointer will be set to the data statement immediately following the label. For example, imagine your program contained the lines:

```
0500 Room15:
0510 DATA "LARGE SNAKE"
```

If elsewhere in your program you had lines:

```
0800 RESTORE'DATA("ROOM15")
0810 READ ANIMAL$
```

the variable ANIMAL$ would now be set to "LARGESNAKE".

If you try to restore the pointer to a label which you did not use in your program, or if the given label does not have a data statement as the very next statement, the program will halt and the appropriate error message will be printed.

Addresses 251-252 serve as the data pointer, while address 253 serves as a flag. When you type RESTORE, the data pointer is set to point to the first data statement in our program, and address 253 becomes 9, and remains such until the next RESTORE command is executed.

RESTORE'LABEL searches through the name table (see issue #4) for the desired label. Once it finds the name, RESTORE'LABEL then checks whether this name is really a label, not the name of a variable, procedure, function etc.. Finally it checks whether the next statement after the label is a data statement. If it is, the data pointer is set to the address of that data statement, and the flag at 253 is set to 4.

```
proc restore'label(name$) closed
 label'length:=len(name$)
 found:=false
 address:=peek(60)+256*peek(61)
 name'end:=peek(62)+256*peek(63)
 counter:=0
 repeat
  this'length:=peek(address)
  if this'length=label'length then
   found:=true
   for i:=1 to label'length do
    if peek(address+i)<>ord(name$(i)) t
    hen found:=false // wrap line
   endfor i
  endif
  if found=false then
   address:+this'length+1
   counter:+1
  endif
 until found=true or address>name'end
 if found=false then
  print
  print "Error! Label ",name$," not fou
  nd." // wrap line
 else
  address:=peek(58)+256*peek(59)+counte
  r*5 // wrap line
  if peek(address)<>19 then
   print
   print "Error! Label ",name$," not fo
   und." // wrap line
   end
  else
   data'address:=peek(address+1)+256*pe
   ek(address+2)+5 // wrap line
   if peek(data'address+3)=170 then
    poke 251,data'address mod 256
    poke 252,data'address div 256
    poke 253,4
   else
    print
    print "Error! Label ",name$," not f
    ollowed by" // wrap line
    print "data statement."
    end
   endif
  endif
 endif
endproc restore'label
```

## C64 GRAPHICS WITH COMAL BOOK

QUESTION: Is the COMMODORE 64 GRAPHICS WITH COMAL from Reston Publishing ready for sale now? What is its price? Is there a disk with it?

ANSWER: Yes, the book finally was released the last week of January 1985. We have 100 copies in stock. List price is $17.95 but we will be selling it for only $14.95 (plus shipping $2 as usual). There is a matching disk available direct from Reston ($19.95 per disk), but we haven't seen it.

# COMPUTER LOG SYSTEM IN COMAL 2.0

by Robert N Nelson

Use this program to keep a disk log of computer use. It uses the 'free' area 740-774 to store time and date. When you run it in logon mode it retrieves them and updates the disk file with logon and logoff time and date, automatically correcting for nightowls who work past midnight. Use INITIALIZE'FILE to set up the log file. To prevent accidental use, it is setup to only be accessed from direct mode.

```
// delete "0:log'program"
// save   "0:log'program"
// Created by Robert N. Nelson
PAGE
USE system
ZONE 0
PRINT TAB(10),"Welcome to logon/off"
PRINT
PRINT "Use Military Time (0-23 hours)"
x:=used'yet
DIM dateon$ OF 9, dateoff$ OF 9
DIM timeon$ OF 8, timeoff$ OF 8
DIM l$ OF 80
DIM activity$ OF 40
IF used'yet THEN
  PRINT AT 5,34: "2"
ELSE
  PRINT AT 5,34: "1"
ENDIF
REPEAT
  PRINT AT 5,1: "enter 1 for logon,";
  INPUT AT 0,0,1: "2 for logoff  ": q$
UNTIL q$ IN "12"
CASE q$ OF
WHEN "1"
  logon
WHEN "2"
  logoff
ENDCASE
STOP
//
PROC logon
  USE system
  INPUT AT 7,1,9: "enter date in dd-mmm
  -yy >": dateon$ // wrap line
  INPUT AT 8,1,8: "enter time in hh:mm
       >": timeon$ // wrap line
  settime(timeon$)
  timeon$:=gettime$(1:8)
  FOR i:=1 TO 9 DO
    c:=ORD(dateon$(i))
    POKE 750+i,c
  ENDFOR i
  FOR i:=1 TO 8 DO
    c:=ORD(timeon$(i))
    POKE 740+i,c
  ENDFOR i
ENDPROC logon
//
PROC logoff
  PAGE
  PRINT "logoff program"
  dateon$:=SPC$(9); timeon$:=SPC$(8)
  PRINT AT 5,1: " place log disk in dri
  ve, <cr> " // wrap line
  WHILE KEY$=CHR$(0) DO NULL
  IF file'exists("logfile") THEN
    OPEN FILE 1,"logfile",APPEND
  ELSE
    OPEN FILE 1,"logfile",WRITE
  ENDIF
  // get logon data
  retrieve'data
  timeoff$:=gettime$(1:9)
  dateoff$:=dateon$
  IF timeoff$(1:2)<timeon$(1:2) THEN ne
  xt'day // wrap line
  activity$:=SPC$(40)
  INPUT AT 10,1,40:"Enter session activ
  ity <40 char long":activity$//wrapline
  PRINT FILE 1: " logon at"," ",dateon$
  ," ",timeon$," " // wrap line
  PRINT FILE 1: activity$
  PRINT FILE 1: " logoff at ",dateoff$,
  " ",timeoff$ // wrap line
  CLOSE FILE 1
  PRINT "logon at ",dateon$;timeon$
  PRINT "logoff at ",dateoff$;timeoff$
  examine'log
ENDPROC logoff
//
PROC next'day
  // update if time crosses midnight
  // this proc automatically takes
  // account of month lengths and
  // of leap years
  //
  DIM month$ OF 3
  day:=VAL(dateoff$(1:2))
  month$:=dateoff$(4:6)
  year:=VAL(dateoff$(8:9))
  newday:=day+1
  dateoff$(1:2):=STR$(newday)
  CASE month$ OF
  WHEN "jan"
    IF newday=32 THEN dateoff$(1:6):="0
    1-feb" // wrap line
  WHEN "feb"
    IF newday=30-SGN(year MOD 4) THEN d
    ateoff$(1:6):="01-mar" // wrap line
    //takes care of normal & leap years
  WHEN "mar"
    IF newday=32 THEN dateoff$(1:6):="0
    1-apr" // wrap line
```

```
  WHEN "apr"
    IF newday=31 THEN dateoff$(1:6):="0
    1-may" // wrap line
  WHEN "may"
    IF newday=32 THEN dateoff$(1:6):="0
    1-jun" // wrap line
  WHEN "jun"
    IF newday=31 THEN dateoff$(1:6):="0
    1-jul" // wrap line
  WHEN "jul"
    IF newday=32 THEN dateoff$(1:6):="0
    1-aug" // wrap line
  WHEN "aug"
    IF newday=32 THEN dateoff$(1:6):="0
    1-sep" // wrap line
  WHEN "sep"
    IF newday=31 THEN dateoff$(1:6):="0
    1-oct" // wrap line
  WHEN "oct"
    IF newday=32 THEN dateoff$(1:6):="0
    1-nov" // wrap line
  WHEN "nov"
    IF newday=31 THEN dateoff$(1:6):="0
    1-dec" // wrap line
  WHEN "dec"
    IF newday=32 THEN
      dateoff$(1:6):="01-jan"
      dateoff$(8:9):=STR$(year+1)
    ENDIF
  ENDCASE
ENDPROC next'day
//
PROC examine'log
  l$:=SPC$(80)
  OPEN FILE 1,"logfile",READ
  REPEAT
    INPUT FILE 1: l$
    PRINT l$
  UNTIL EOF(1)
  CLOSE FILE 1
ENDPROC examine'log
//
PROC retrieve'data
  dateon$:=SPC$(9); timeon$:=SPC$(8)
  FOR i:=751 TO 759 DO
    dateon$(i-750):=CHR$(PEEK(i))
  ENDFOR i
  FOR i:=741 TO 748 DO
    timeon$(i-740):=CHR$(PEEK(i))
  ENDFOR i
  PRINT "logon at "," ",dateon$," ",tim
  eon$ // wrap line
ENDPROC retrieve'data
//
PROC initialize'file
  // use to initialize file
  PAGE
  DIM file'name$ OF 16
  PRINT ""137""14""9""20""9""1""12""9""
  26""5" "140""15""7" "134""9""12""5""
  //line above is a continuation line
  INPUT AT 2,1,16: "Enter log-file name
   ": file'name$ // wrap line
  OPEN FILE 1,file'name$,WRITE
  PRINT FILE 1: " Initial Log File "
  CLOSE FILE 1
ENDPROC initialize'file
//
FUNC used'yet
  //returns true if data has been loaded
  z:=0
  FOR i:=741 TO 748 DO z:+PEEK(i)
  FOR i:=751 TO 759 DO z:+PEEK(i)
  IF z>17 THEN
    RETURN TRUE
  ELSE
    RETURN FALSE
  ENDIF
ENDFUNC used'yet
//
FUNC file'exists(name$) CLOSED
  TRAP
    OPEN FILE 78,name$,READ
    CLOSE FILE 78
    RETURN TRUE
  HANDLER
    CLOSE FILE 78
    RETURN FALSE
  ENDTRAP
ENDFUNC file'exists
```

## SUBSTRINGS - EASY

Several people have asked what they could do about the lack of LEFT$, RIGHT$, and MID$ keywords. Those are the keywords that BASIC uses to specify substrings. COMAL has an easier, more flexible method. So, it's not that COMAL is missing the substring keywords - it is that COMAL has a different way to accomplish the same thing (and more). For example:

```
DIM TEXT$ OF 20
TEXT$:="ABCDEFGHIJ"
PRINT TEXT$(1:3) // prints ABC
PRINT TEXT$(5:6) // prints EF
PRINT TEXT$(8:10)// prints HIJ
```

The above can all be done in BASIC easily too. However, COMAL lets you CHANGE any part of a string without affecting the rest of it! BASIC has no direct way of doing this. So - take advantage of it: try this added to the above program:

```
TEXT$(3:4):="XX"
PRINT TEXT$ // prints ABXXEFGHIJ
```

# FIT IT FOR COMAL 0.14

by Randy Finch

Fit'it v1.0 is a program written in COMAL v0.14. It will accept up to 80 x,y points, plot them on a graph, and do a least-squares curve fit for eight different equations. It will plot the curve of your choosing. A screen dump program (included) written in machine language will allow you to print the graph on your Gemini 10 or compatible printer.

To run Fit'it, simply type:

```
LOAD "BOOTFIT" <RETURN>
RUN            <RETURN>
```

This will load a boot program which POKEs a machine language screen dump program to the cassette buffer (address 828) and then CHAINs the main program called FIT'IT. If the screen dump I have written does not work with your printer, you can substitute your own routine in the data statements in the boot program.

The main program will begin with an input phase asking for your list of x,y coordinates. Up to 80 ponts can be entered. Each point is entered as a pair of coordinates. Type the x coordinate and the y coordinate, separated by a comma and followed by a return - for example, if your x coordinate was 10 and your y coordinate was 30, you would enter:

```
10,30   <RETURN>
```

Zeros and negative numbers are not permitted because some of the curve fits require division by the coordinate or by the natural log of the coordinate. When you finish entering our data, enter a 0,0 as your next coordinate, and the input routine will end. You will then be asked if you would like to review your data. If you think you have made a mistake, answer with a Y. The data will be displayed and you will have a chance to delete any of the points that were entered incorrectly. After this phase, you will be asked if you would like to enter more data. If you missed any data, or if you would like to replace any data you deleted, answer Y. You will then be allowed to add data just as you did in the initial phase of data entry. You will be asked over and over for deletions and additions until you answer N to the questions. Next, you will be asked if you want a hardcopy of the data you just input. Answer Y if you do. This will end the initial phase of the program.

The next phase of the program is the information phase. First, you will be asked to label the x-axis of your graph. This label can be up to 30 characters long. Next, you will enter the y-axis label, which can be up to 23 charaters long. You will then be asked for six pieces of information about the graph set-up. In order, these are:

1. Smallest value of x on the graph, x-mininum
2. Largest value of x on the graph, x-maximum
3. The distance between tic marks in the x-axix, x-tic width

4-6. Same as 1-3, except for the y-axis

You will now be asked if you want a hardcopy of the equasions that are about to appear. If you answer Y, the equations will both be displayed on your screen, and sent to your printer. If you answer N, they will be displayed only to the screen.

Next, the program moves to its output phase. You will be shown eight equations on the screen along with the corresponding values of R-square. The closer this value is to unity the better the curve fit is. The equations you will see have the following forms:

1. Y=a+bX
2. Y=ae↑(bX)
3. Y=ab↑X
4. Y=aX↑b
5. Y=a+b*ln(X)
6. Y=b/(a+X)
7. Y=bX/(1+aX)
8. Y=a+b/X

You will be asked which equation you would like to see plotted. Choose a number between 1 and 8 or a 0 if you don't want ot see any. The curve you selected will be plotted on the screen. If you want a hardcopy of the graph, press P while viewing the graph. If you want to see the equations again press <RETURN>. (Note: Do not be alarmed if the response to pressing P or <RETURN> is

slow. The program plots the curve based on every point of X along the X-axis. Sometimes the plot will rurn off the graph and even though it looks like the plot is finished, the program is still churning out numbers. If you press P or <RETURN> and nothing happens, just wait, because the key you pressed is stored in the keyboard buffer.) You will be able to keep viewing and printing graphs until you enter a 0 for the equation number to view. At this point you will have an opportunity to input a completely new set of data or end the program.

## GRAPHIC SCREENS IN COMAL 0.14

by Phyrne Bacon

Phyrne Bacon has previously submitted some beautiful graphics art programs. Now she has provided us with several programs to SAVE the graphics screens to disk and (of course) to retrieve these screens any time later. She even has provided a program that can load in a graphics screen saved from Commodore LOGO. A set of five programs are included on TODAY DISK #6 for your use. SAVE HIRES COLOR saves a color graphics screen in about 35 seconds. These screens can be loaded by her VIEW COLOR programs as well as directly into COMAL 2.0 Cartridge with its built in LOADSCREEN command (they are upward compatible!).

The SAVE program first stores the picture under the file name HRG.DEFGHIJKLMNO and after saving it asks you for the name you would like to use. Then it just renames the file for you. This arrangement allows you to chain to this SAVE program from another program without disturbing the color nybbles used in multicolor hires. The file produced is 36 blocks from SETGRAPHIC 0 and 40 blocks from SETGRAPHIC 1.

Getting the screen images back again takes only about 30 seconds with VIEW COLOR. It can retrieve pictures stored with the SAVE program as well as COMAL 2.0 pictures stored with its built in SAVESCREEN command. The VIEW LOGO program will retrieve a LOGO picture which is stored in two separate files, one 33 blocks long and the other 5 blocks long (she even incuded a LOGO picture named BOX&CIRCLE).

This program is on TODAY DISK #6.

## MORE ON THE :+ BUG

by Ian MacPhedran

In COMAL TODAY #4 Kevin Quiggle demonstrated a bug in the :+ command. (This problem also exists in the :- command). This bug has another feature which should be noted. In a regular assignment statement, you cannot use variables which have not been defined. However, the assign and increment (decrement) statement does not check if the variable being incremented (decremented) has been defined or not. This could lead to strange results. For example, if the name table assigns the variable to a location previously used by another variable (this could happen in the immediate mode, after a NEW) the variable takes on the value of the previous variable, and then is incremented (decremented). If in a program, the variable keeps its value from previous runs. This means that after the first run, instead of having 0 to begin the second run, the variable ha the last value it had from the first run. This may not be desired. However, this problem is eliminated in COMAL 2.0.

To see how the bug works try the following program, and RUN it three or four times. Note how the value of TEST increases each time, although it should be the same each time.

```
NUM:=4
TEST:+NUM
PRINT TEST
```

## AUTOMATED PROC & FUNC LIBRARIAN

by Kevin Quiggle

One of COMAL's strong points is its ability to merge program segments from disk. Why reinvent the wheel every time you write a program? Kevin Quiggle collected nearly 150 useful routines on one disk and then documented them all. This book/disk set is LIBRARY OF FUNCTIONS & PROCEDURES. And now to make it EASY to merge all those routines into your programs he has provided the AUTO LIBRARIAN program which is included on TODAY DISK #6. You must have the LIBRARY disk to be able to use AUTO LIBRARIAN. **See page 1 for a special on the LIBRARY set.**

# ANOTHER LOOK AT COMAL 0.14 TOKENS by Ian MacPhedran

After reading John H McCoy's article in COMAL TODAY #5 on tokenization in COMAL, I went thru version 0.14 in search of more tokens. Below is a list of the tokens which Mr McCoy published, as well as those which I have identified. I have also looked at some of the internal structure of version 2.0 statements. It is interesting to note that 0.14 starts line numbers with a zero offset (ie, line 10 is store as line 10) while 2.0 has an offset of 10000 (line 10 is stored as 10010). Both versions use many of the same tokens. This helps with compatibility, as well as with compiling a list such as this for the cartridge.

Two tokens of interest are $89 and $FF. The former ($89) does not translate out as a keyword, but is included in all the structured statements. The latter ($FF) does not translate out either, but acts as a flag to signify that the following token is to be treated as a graphics keyword. A partial list of the graphics keywords follow.

| | | | | | |
|---|---|---|---|---|---|
| $00 | REM | $2C | NUM'LOP'EQ (=) | $58 | SPC$ |
| $01 | | $2D | STR'LOP'EQ (=) | $59 | SQR |
| $02 | | $2E | NUM'LOP'LE (<=) | $5A | TAN |
| $03 | | $2F | STR'LOP'LE (<=) | $5B | TIME |
| $04 | | $30 | NUM'LOP'GT (>) | $5C | EOD |
| $05 | | $31 | STR'LOP'GT (>) | $5D | EOF |
| $06 | | $32 | NUM'LOP'NE (<>) | $5E | |
| $07 | STMNT'SEP (;) | $33 | STR'LOP'NE (<>) | $5F | PRINT'START |
| $08 | | $34 | NUM'LOP'GE (>=) | $60 | PRINT'CRLF |
| $09 | | $35 | STR'LOP'GE (>=) | $61 | PRINT'END |
| $0A | | $36 | IN | $62 | USING |
| $0B | | $37 | NOT | $63 | TAB |
| $0C | | $38 | AND | $64 | PRINT'NUMB |
| $0D | ARRAY'COMMA | $39 | OR | $65 | PRINT'STRING |
| $0E | | $3A | ASGN'REAL | $66 | PRINT'TAB |
| $0F | | $3B | ASGN'INTEGER | $67 | PRINT'SPACE |
| $10 | | $3C | ASGN'STRING | $68 | IF |
| $11 | | $3D | ADD'REAL'DEST | $69 | THEN |
| $12 | | $3E | ADD'INTEGER'DEST | $6A | THEN'ONE'LINE |
| $13 | ARRAY'SOURCE'PAREN | $3F | ADD'STRING'DEST | $6B | |
| $14 | | $40 | SUB'REAL'DEST | $6C | |
| $15 | | $41 | SUB'INTEGER'DEST | $6D | ELIF |
| $16 | | $42 | | $6E | ELSE |
| $17 | | $43 | TRUE | $6F | ENDIF |
| $18 | | $44 | FALSE | $70 | PROC |
| $19 | SUBSTRING'PAREN | $45 | ZONE'READ | $71 | |
| $1A | ARRAY'DEST'PAREN | $46 | ZONE'SET | $72 | PARM'REAL |
| $1B | | $47 | PAREN (()) | $73 | PARM'INTEGER |
| $1C | | $48 | ABS | $74 | PARM'STRING |
| $1D | | $49 | ORD (ASC?) | $75 | REF'PARM'REAL |
| $1E | | $4A | ATN | $76 | REF'PARM'INTEGR |
| $1F | SUBSTRING'COLON | $4B | CHR$ | $77 | REF'PARM'STRING |
| $20 | POSITIV (+) | $4C | COS | $78 | REF'PARM'REAL'ARR |
| $21 | NEGATIV (-) | $4D | ESC | $79 | REF'PARM'INT'ARR |
| $22 | EXPONTN ( ) | $4E | EXP | $7A | REF'PARM'STR'ARR |
| $23 | DIVISN (/) | $4F | INT | $7B | PARM'PAREN |
| $24 | MULTN (*) | $50 | | $7C | PARM'PAREN'CLOSED |
| $25 | DIV | $51 | LEN | $7D | ENDPROC |
| $26 | MOD | $52 | LOG | $7E | PASS |
| $27 | ADDTN (+) | $53 | RND (1 NUM) | $7F | EXEC |
| $28 | CONCAT (+) | $54 | RND (2 NUM) | $80 | STATUS$ |
| $29 | SUBTN (-) | $55 | RND'COMMA | $81 | |
| $2A | NUM'LOP'LT (<) | $56 | SGN | $82 | FOR'REAL |
| $2B | STR'LOP'LT (<) | $57 | SIN | $83 | FOR'INTEGER |

```
$84   FOR'FROM
$85   FOR'TO
$86   FOR'STEP
$87   FOR'DO
$88   FOR'DO'ONE'LINE
$89   COMPARISION???
$8A   ENDFOR'REAL
$8B   ENDFOR'INTEGER
$8C   DIM
$8D   DIM'REAL
$8E   DIM'INTEGER
$8F   DIM'STRING
$90   DIM'COLON
$91   DIM'COMMA
$92   DIM'PAREN
$93   DIM'STRING'LEN
$94   DIM'SEP
$95   REPEAT
$96   REPEAT'END
$97   WHILE
$98   DO'WHILE
$99   DO'WHILE'ONE'LINE
$9A   ONE'LINE'ENDWHILE
$9B   ENDWHILE
$9C   LABEL
$9D   GOTO
$9E   DELETE
$9F   END
$A0   STOP
$A1   CASE
$A2   CASE'OF
$A3
$A4   WHEN
$A5   CASE'SEP
$A6
$A7   CASE'TERMIN
$A8   OTHERWISE
$A9   ENDCASE
$AA   DATA'START
$AB   DATA'NXT
$AC   DATA'REAL
$AD   DATA'INT
$AE   DATA'STR
$AF   READ'START
$B0   READ'REAL
$B1   READ'INT
$B2   READ'STR
$B3   READ'END
$B4   RESTORE
$B5   INPUT'START
$B6   INPUT'PROMPT'?
$B7   INPUT'PROMPT'STR
$B8   INPUT'REAL
$B9   INPUT'INT
$BA   INPUT'STR
$BB   INPUT'END
$BC
$BD   SELECT
$BE   OUTPUT
$BF   TRAP
$C0   ESC
$C1   ESC'TRUE  (+)
$C2   ESC'FALSE (-)
$C3   WRITE
$C4   WRITE'NUM
$C5   END'DIM
$C6   WRITE'STR
$C7   WRITE'FILE
$C8   WRITE'RECORD
$C9   WRITE'OFFSET
$CA   WRITE'COLON
$CB
$CC
$CD
$CE   CLOSE
$CF   CLOSE'FILE
$D0
$D1
$D2   CHAIN
$D3   CHAIN'DEFAULT'DEV (8)
$D4   CHAIN'END'LINE
$D5   CHAIN'SPECIFY'DEV
$D6
$D7
$D8   PEEK
$D9
$DA   UNTIL
$DB   OPEN
$DC   FILE
$DD   FILE'NAME
$24   CLEAR
$25   DEFINE
$DE   FILE'TYPE'READ
$DF   FILE'TYPE'WRITE
$E0   FILE'TYPE'APPEND
$E1   FILE'TYPE'RANDOM
$E2   UNIT
$E3   UNIT'SEC'ADD
$E4   POKE
$E5
$E6
$E7
$E8
$E9   FUNC'REAL
$EA   FUNC'INTEGER
$EB
$EC   ENDFUNC'REAL
$ED   ENDFUNC'INT
$EE
$EF   RETURN'REAL
$F0   RETURN'INT
$F1
$F2
$F3
$F4   NULL
$F5
$F6
$F7   SPRITECOLLISION
$F8   DATACOLLISION
$F9   KEY$
$FA   GETCOLOR
$FB
$FC
$FD
$FE
$FF   GRAPHICS'FLAG

GRAPHICS CODES FOLLOW:
$16   SETGRAPHIC
$19   SETTEXT
$1A   FRAME
$1C   PLOT
$1E   PENCOLOR
$20   BACKGROUND
$22   PLOTTEXT
$27   INDENTIFY
$29   SPRITECOLOR
$2B   SPRITEPOS
$2D   SPRITESIZE
$2F   HIDESPRITE
$31   SPRITEBACK
$33   MOVETO
$35   DRAWTO
$37   FILL
$39   FORWARD
$3B   SETXY
$3C   LEFT
$3E   SETHEADING
$40   PRIORITY
$42   PENUP
$44   PENDOWN
$46   BORDER
$48   RIGHT
$4A   BACK
$4C   SPLITSCREEN
$4E   SHOWTURTLE
$50   TURTLESIZE
$52   FULLSCREEN
$53   HIDETURTLE
$54   HOME

      OPERANDS
$01   REAL'CONST
$02   INT'CONST
$03   STR'CONST
$04   REAL'SOURCE
$05   INT'SOURCE
$06   STR'SOURCE
$07   REAL'DEST
$08   INT'DEST
$09   STR'DEST
$0A   REAL'ARRAY
$0B   INT'ARRAY
$0C   STR'ARRAY
```

# GOODBYE GOTO



In March 1968 the science of programming began. In that month Edsger W Dijkstra published a letter titled, "GO TO STATEMENT CONSIDERED HARMFUL" in the Communications of the ACM, the most prestigious of the professional computer journals.

In his letter, Dr. Dijkstra made several radical statements:

"The quality of programmers is a decreasing function of the density of GO TO statements in the programs they produce."

"The GO TO statement should be abolished from all 'higher level' programming languages (i.e. everything except, perhaps, plain machine code)."

"The <GO TO statement> is too much an invitation to make a mess of one's program."

Unfortunately BASIC is such a poorly designed language that almost any program requires many GOTO's. Continued use of such a language, especially when it is the first language learned, is harmful as it reinforces the bad habit of using GOTO statements. Fortunately BASIC is not the only language available. COMAL can free you from the dreaded GOTO. Once you realize that there are better things in life than the GOTO statement, you won't even miss it! You may not remember it, but it took you a while to learn to write a problem using GOTO statements. What I hope to do in this article is to show you how to think about your problem like you did before you met BASIC and to show you that you can express that thought directly in COMAL without having to translate it into GOTO statements. I intend to accomplish this goal by examining the typical functons of GOTO statements and showing proper COMAL code for those functions.

A GOTO statement in a program must either refer backward in the program (to a lower line number) or forward. (A GOTO that goes to its own line is a backward reference since it goes back from the end of that line to its beginning.) A backward reference should only be used to form a loop while a forward reference should only be used to skip some code. (There are people who use GOTO statements for other purposes but they need more help than an article like this can offer.) Le's look at forward and backward GOTO's in more detail.

There are many times you may want your program to repeat some action. If you know in advance how many times you want the action done, you would use a FOR-NEXT loop (in either BASIC or COMAL.) For example, to print the first 10 numbers and their square roots you would code:

```
100 for x = 1 to 10
110 print x; tab(10); sqr(x)
120 next x
```

Suppose, however, you don't know how many times you want something done. A typical example is waiting for the user to press any key. In BASIC this is often code as:

```
100 geta$:ifa$=""thengoto100
```

The backward loop would be more clearly visible if we split the two statements onto separate line:

```
100 geta$
101 ifa$="" thengoto100
```

What we really want to accomplish is to look at the keyboard and to keep (repeat) looking at it until a key is pressed. Such a loop can be coded in COMAL in two ways:

```
100 repeat
101 until key$ <> ""
```

or

```
100 while key$ = "" do null
```

Again, unfortunately, I do not have room here to describe all aspects of the REPEAT, WHILE, and FOR statements. Look them up in your COMAL HANDBOOK and explore the examples on the various COMAL disks. Use these statements instead of GOTO's to form loops and you will find your programs easier to write, read, and debug.

Now, what about GOTO statements that go forward to a higher line number? They are used in BASIC to skip sections of code.

In COMAL you should use the IF-THEN-ELSE or CASE statements. Let's take a simple, if useless, example. Suppose your program has asked a person if they prefer cats or dogs and you have set A$ to be either "C" or "D" to represent their answer. You now wish to give some information about their favorite animal. In BASIC you might code:

```
100 ifa$="c"thenprint"cats catch mice.":
goto200
110 print"dogs bite mailmen."
200 rem ... rest of program
```

In COMAL you might code this as:

```
100 if a$ = "c" then
110  print "cats catch mice."
120 else
130  print "dogs bit mailmen."
140 endif
```

Although the COMAL version takes more lines, it requires about the same amount of storage and runs slightly faster than the BASIC program. It is also much easier to read and understand!

If you want to use a menu in your program, you might print the choices on the screen and accept a reply. In BASIC you might then say:

```
100 on r goto 200,300,400
110 print "bad reply",r
200 print "first option"
210 goto 500
300 print "second option"
310 goto 500
400 print "third option"
500 rem ... rest of program
```

In COMAL you would use the CASE statement:

```
100 case r of
110 when 1
120  print "first option"
130 when 2
140  print "second option"
150 when 3
160  print "third option"
170 otherwise
180  print "bad reply",r
190 endcase
```

These examples are, of course, simpler than real life but the techniques work equally well with real programs. Look up the IF-THEN-ELSE and CASE statements in your COMAL HANDBOOK and study the examples on the various COMAL disks.

If you already have your COMAL 2.0 Cartridge you should also study the LOOP, EXIT, and the TRAP-HANDLER statements.

Finally, I would liek to leave you a caution. Using the above structures and avoiding GOTO statements will not hurt you and will probably help most people a great deal. It is still possible, though more difficult, to screw up a program even if it doesn't have a single GOTO in it. This becomes more important when you try to convert programs from an unstructured language like BASIC to a structured language like COMAL or PASCAL. Dr. Dijkstra warned, "The exercise to translate an arbitrary flow diagram more or less mechanically into a jump-less one, however, is not to be recommended. Then the resulting flow diagram cannot be expected to be more transparent than the original one." In other words, you still have to use good common sense and approach your program in a reasonable manner. Using a language like COMAL gives you better tools to work with but doesn't guarantee you won't build trash or hit your thumb with a hammer.

## COMAL 2.0 IS COMPATIBLE WITH 0.14

QUESTION: My impression of the cartridge was that it is compatible with the disk version 0.14, however, I find that when I try to either LOAD or F7 RUN one of my programs I wrote using COMAL 0.14, I get the message NOT COMAL PROGRAM FILE. I also get the same message when trying to LOAD programs from the UTILITES DISK and TODAY DISK #1.

ANSWER: The COMAL is compatible - the LOAD / SAVE files are not. You must LIST a program to disk from COMAL 0.14 and then use ENTER from COMAL 2.0. Then the files are compatible. From COMAL 0.14 store your program to disk like this:

```
LIST "0:NAME.L"
```

Then while in COMAL 2.0, you retrieve it like this:

```
ENTER "0:NAME.L"
```

**COMAL ORDER FORM - COMAL USERS GROUP, U.S.A., Ltd, 5501 Groveland Ter, Madison, WI 53716**
phone: 608-222-4432

Name:_____ (3/85)

Street:_____

City:_____ State:_____ ZIP:_____

**NOTE:** Pay by check, money order in US Dollars drawn on US Bank. Canada Postal US Dollar Money Orders are OK. VISA/MasterCard: PRINT card #, exp date below with your signature:

VISA / MC number:_____exp date:_____

Signature:_____

**Are you a COMAL TODAY subscriber? YES / NO / NEW** (If YES subscriber# required:_____)
(If you subscribe now circle NEW and get the subscribers prices right away)

| QNTY | AMOUNT | ITEM DESCRIPTION and LIST PRICE (**all disks Commodore 1541 format**) |
|---|---|---|
| | | **SYSTEMS:** |
| [ ] | _____ | Deluxe Cartridge Package, $128.90 plus $3 shipping |
| [ ] | _____ | COMAL 2.0 Cartridge, $99.95 plus $2 shipping (no manuals or disks) |
| [ ] | _____ | C64 COMAL 0.14 Starter Kit, $29.95 plus $2 shipping |
| [ ] | _____ | IBM PC COMAL from Mytech / Sweden, $200 (MS-DOS 2.0 format disk) |
| | | **SUBSCRIPTIONS:** |
| [ ] | _____ | **COMAL TODAY newsletter**, $14.95 for 6 issues plus $2 each after that |
| | | How many issues? _____ (renewals must include subscriber# _____) |
| [ ] | _____ | TODAY DISK subscription, $49.90 for six disks plus $5 each after that |
| | | How many disks? _____ ==> State starting disk num:_____ |
| | | **BOOKS: (shipping $2 each)(matching disks are only $4.95 to subscribers)** |
| [ ] | _____ | COMAL HANDBOOK (second edition, second printing), $18.95 |
| [ ] | _____ | optional matching disk, $19.95 ($4.95 to subscribers) |
| [ ] | _____ | FOUNDATIONS IN COMPUTER STUDIES WITH COMAL (new second edition), $19.95 |
| [ ] | _____ | optional matching disk, $19.95 ($4.95 to subscribers) |
| [ ] | _____ | STRUCTURED PROGRAMMING WITH COMAL, $26.95 (new price) |
| [ ] | _____ | optional matching disk, $19.95 ($4.95 to subscribers) |
| [ ] | _____ | BEGINNING COMAL, $22.95 (**new price**)- 2nd printing due here by about Apr 1 |
| [ ] | _____ | optional matching disk, $19.95 ($4.95 to subscribers) |
| [ ]>> | _____ | COMMODORE 64 GRAPHICS WITH COMAL, $14.95 special (list price $17.95) |
| [ ] | _____ | COMAL FROM A TO Z - $6.95 (now in its FOURTH printing) |
| [ ] | _____ | CAPTAIN COMAL GETS ORGANIZED, $19.95 ($14.95 to subscribers) |
| [ ] | _____ | COMAL WORKBOOK, $6.95 (70 full size pages) |
| [ ] | _____ | COMAL LIBRARY OF FUNCTIONS AND PROCEDURES, $19.95 ($14.95 to subsribers) |
| [ ] | _____ | GRAPHIC PRIMER, $19.95 ($14.95 to subscribers) |
| [ ] | _____ | CARTRIDGE GRAPHICS AND SOUND, $9.95 (also part of Deluxe Cartridge Pak) |
| [ ]>> | _____ | COMAL 2.0 PACKAGES, $19.95 (book & disk) includes COMSYMB and COMAL monitor |
| | | **DISKS: (NOTE: all disks are only $9.75 each to subscribers!).** |
| [ ]>> | _____ | 19 DISK SET, $94.05 (about 1000 programs for COMAL 0.14) |
| [ ]>> | _____ | BEST OF COMAL, $19.95 ($9.75 to subscribers)(includes Disk Data Base) |
| [ ] | _____ | C64 COMAL SAMPLER, $19.95 ($9.75 to subscribers)(part of Starter Kit) |
| [ ] | _____ | AUTO RUN DEMO DISK, $19.95 ($9.75 to subscribers)(part of Starter Kit) |
| [ ] | _____ | COMAL TUTORIAL DISK, $19.95 ($9.75 to subscribers)(part of Starter Kit) |
| [ ]>> | _____ | Bricks TUTORIALS (2 sided BEGINNERS disk!), $19.95 ($9.75 to subscribers) |
| [ ]>> | _____ | Modem Disk $14.95 (9.75 to subscribers)(for COMAL 0.14 & 2.0). |
| [ ] | _____ | UTILITY DISK, $19.95 ($9.75 to subscribers) |
| [ ] | _____ | TODAY DISK, $14.95 **>>>> Which one(s)** (1-6): _____ |
| [ ] | _____ | USER GROUP DISK, $9.75 **>>>> Which one(s)** (1-9): _____ |
| [ ] | _____ | CARTRIDGE DEMO DISKS, ($9.75 to subscribers) **>> Which** (1-4): _____ |
| [ ] | _____ | OTHER: _____ |
| [ ]>> | _____ | KEYBOARD OVERLAY for C64 COMAL 0.14 - CHEATSHEET, $3.95 (plus $1 handling) |
| | ========== | |
| | $_____ | <==TOTAL COST ITEMS |
| | $_____ | <==ADD SHIPPING (minimum $2) |
| | $_____ | <==TOTAL PAID (No COD or PO's) |

Yes, I want COMAL TODAY. Give me _____ issues.
($14.95 for 1st 6, $2 each after that)

This is a: [ ] New Subscription
[ ] Renewal
My subscriber # is:_____

Name:_____

Address:_____

City:_____State:_____ZIP:_____

[ ] Check enclosed [ ] Visa [ ] MasterCard

Card # _____expires_____

Signature:_____

From: _____

_____

_____

PLACE
STAMP
HERE

TO: COMAL USERS GROUP, USA, LIMITED
5501 GROVELAND TERRACE
MADISON, WI 53716-3251

**Today Disk 6**
**COMAL 0.14 Side**

"HI"
"MENU"
">---PROGRAMS---<"
"1525'SCREEN'DUMP"
"APRILFOOL"
"ART"
"BOOT'DATA'BASE"
"BOOT'FIT'IT"
"BOUNCE'BALL.14"
"BOWLING'SCORE"
"COMAL'KEYPAD.14"
"CREATE'LIB'DATA"
"DATA'BASE'MGR"
"DBASE14"
"DRAW'DAISY"
"EXPAND'RAM"
"FIND"
"FIT'IT"
"GRADES"
"LABEL"
"LIBRARIAN"
"MAGIC"
"MAIL'LABEL"
"MICROSCOPE QUIZ"
"MINDY'S-DEMO"
"PINWHEEL"
"PLAYNET"
"PRINT'2'COL'DIR"
"PRINT'CALENDAR"
"PRINT'DIRECTORY"
"QUADRATIC'ROOT"
"SAVE'COLOR'DATA"
"SAVE'COLOR'POKES"
"SAVE'ERROR'FILE"
"STAR'80"
"VIEW'COLOR'DATA"
"VIEW'COLOR'POKES"
"VIEW'LOGO"
">--PROCEDURES--<"
">-&'FUNCTIONS--<"
"BUFFER.PROC"
"DIR.L"
"FILE'EXISTS.FUNC"
"FX-80'CMDS.PROC"
"JOYSTICK.PROC"
"LOAD'OBJ.PROC"
"PADDLE.PROC"
"PLOTTEXT.PROC"
"REPEAT'KEY.PROC"
"RESTORE'LBL.PROC"
">--DATA'FILES--<"
"---ERROR-MESS---"
"BOX&CIRCLE.PIC1"
"BOX&CIRCLE.PIC2"
"COMDMPMOD.MEM"
"DANCE.DAT"
"HRG.WORLD'MAP"
"LIB.DAT"
"LOAD/SAVE.MEM"
"MICROSCOPE.DAT"
"MUSIC'PLAYER"
"PHONE'DATA"
"PHONE'DICT"

**Today Disk 6**
**COMAL 2.0 Side**

"HI"
">---PROGRAMS---<"
"BATTLESHIP"
"BOUNCE'BALL"
"CALCULART"
"CHECK'ALL'CARTS"
"CLASS'LABELS"
"COMAL'KEYPAD'2.0"
"COMAL'CLOCK"
"CREATE'SILICON"
"DISASSEMBLER"
"DOG/CAT"
"DOODLE'TO'2.0"
"DRAW'MOLECULES"
"F-KEY OVERLAY"
"FIT'IT'2.0"
"GEMINI'DIR'PRINT"
"GRADING"
"HEX'DUMP"
"LABEL'IT"
"LETTER'SHELL"
"LIGHTPEN'DEMO1"
"LIGHTPEN'DEMO2"
"LOG'PROGRAM"
"MAGIC'VOICE'DEMO"
"POLAR'ROSES"
"POSTER'DOWN"
"POSTER'RIGHT"
"SHADOW'LETTERS"
"SHELL'SORT"
"SONG'EDITOR"
"SONG'PLAYER"
">---EXTERNAL---<"
"EXT.GTR'GRADE"
"EXT.SEM'GRADE"
">--PROCEDURES--<"
"PROC.BUFFER"
"PROC.FX-80'CMDS"
"PROC.GEM'BIGDUMP"
"PROC.GEMINI'DUMP"
"PROC.MAGIC'VOICE"
"PROC.MODEM'WORK"
"PROC.OLIVETIDUMP"
"PROC.REPEAT'KEYS"
">--DATA'FILES--<"
"BAT.0.14'TO'2.0"
"BAT.DUAL'STUFF"
"BAT.EDIT'PROG'S"
"BAT.NORMAL"
"DAT.SILICON"
"HRG.FRONT'PAGE"
"LOGFILE"
"SNG.POLKA/ACC"
"SNG.POLKA/STR"

**User Group 7**

"BOOT C64 COMAL"
"C64 COMAL 0.14"
"---ERROR-MESS---"
"HI"
">--DATA FILES--<"
">-DO NOT LOAD--<"
"ADV GUESS IT"
"AMORT"
"GUESS IT"
"GUESS IT INST"
"LANGUAGE NOTES"
"MICROSCOPE.DAT"
">---EDUCATION--<"
"BOOT GUESS IT"
"CHILL"
"GEOMETRY"
"GRADES"
"HUMIDITY"
"MICROSCOPE QUIZ"
"PERFECT NUMBERS"
">-----GAMES----<"
"BATTLESHIP"
"BOUNCE'BALL"
"CODEMAKER"
"DOTS"
"MASTERMIND'WORD"
"SCORE KEEPER"
">-APPLICATIONS-<"
"BIWEEKLY SAVINGS"
"BOWLING SCORE"
"COLLEGE ED SAVIN"
"FIND"
"GEM10X'LISTER"
"LOAN"
"LOAN'CHART"
"VALUE OF INVESTM"
">----DEMOS-----<"
"2'SINE"
"ART'COLOR"
"BEEP'2"
"DAISY COLOR DEMO"
"FAST'RECTAN"
"GREEN BOXTREE"
"OBELLS"
"ONE'MORE'CIRCLE"
"OVALS"
"PINWHEEL"
"POLAR DAISY"
"QUAD"
"RAINBOWS"
"ROTATED OVALS"
"SPIROGRAPH'NEW"
"STAR 80"
"TANGENT CIRCLES"
"YARN'ART'2"

**User Group 9**
**Side 1**

"MENU"
"ARTICLES"
"LANGUAGE"
"MAINMENU"
"PRG'INST"
"PROGRAMS"
"SIG'NEWS"
"STANDARD"
"ADVERTIS"
"HELP COD"
"WANT'ADS"
"TABLE OF CONTENT"
"COLIN DOES SF"
"FLASH"
"MOVING UP TO 2.0"
"ON BALANS CHAIR"
"OPEN LETTER"
"PARTY QUIZ"
"PLAYNET PART 1"
"PLAYNET PART 2"
"RANDOMTHOUGHTS"
"WILD CARDS"
"YOUR NICHE"
"1541 TIPS"
"APRIL PREVIEW"
"COMMODORE ECKS"
"DIRECTORY ONE"
"DIRECTORY TWO"
"OTHER GROUPS"
"OUR GROUP"
"OUR LIBRARY"
"OUR NEWSLETTER"
"OUR OFFICERS"
"PREZ PAGE"
"TO N/L EDITORS"
"BUTTERFIELD TAPE"
"CAPT COMAL VISIT"
"COMAL CLASSES"
"HOOKING IN"
"LOGO & PILOT"
"PLAYNET"
"USER GROUP DISKS"
"VIC 20 LIST"
"DATA BASE INST"
"DOODLE LOADER"
"DUMP.1525.BIG"
"JOY CURSOR"
"MAIL LIST"
"NUTCRACKER"
"PRINT SHOP PIX"
"RAVICS'TERM"
"SIMULATE'PLAYNET"

**User Group 9**
**Side 2**

"BOOT COD"
"GO"
"TITLE"
"<<< PROGRAMS >>>"
"<<< BASIC >>>"
"DOODLE LOADER"
"NUTCRACKER"
"RAVICS TERM"
"<< COMAL 0.14 >>"
"BOOT DATA BASE"
"DUMP.1525.BIG"
"SIMULATE PLAYNET"
"<< COMAL 2.0 >>>"
"MAIL'LIST"
"JOY CURSOR"
"<PRINT SHOP PIX>"
"SHELL"
"PALM"
"WAVES"
"BUNNY"
"CLAVIER"
"NOTES"
"SMALL NOTE"
"GARFIELD"
"ODIE"
"GARF.HEAD"
"<< DATA FILES >>"
"<< DON'T LOAD >>"
"8MACHINE LANG"
"LOAD/SAVE.MEM"
"MUSIC.DAT"
"PLAYER.OBJ"
"HRG.COVER"
"DATA'BASE'MGR"
"DBASE14"
"NUTCR2"
"NUTCR3"
"NUTCR4"
"<< DOODLE PIX >>"
"DDCOMAL CALVIN"
"DDCABBAGE PATCH"

**Bricks Tutorial**
**Side 1**

"BOOT"
"C64 COMAL 0.14"
"HI"
"MENU"
"TURTLE'MENU"
"CURSOR"
"CURSOR2"
"EDIT1"
"EDIT2"
"FILL"
"FRAME"
"GETTING STARTED"
"GETTING STARTED2"
"GRAPHICS"
"HEADINGS"
"HELLO1"
"HELLO2"
"HELLO3"
"HELLO4"
"HELLO5"
"HELLO6"
"INPUT1"
"INPUT2"
"LOOPS"
"LOOPS2"
"MOVE"
"PLOTTEXT"
"SIZE"
"STEPS1"
"STEPS2"
"STEPS3"
"TAKING CONTROL"
"TURNS"
"TURTLE"
"TURTLE2"

**Bricks Tutorial**
**Side 2**

"BOOT"
"C64 COMAL 0.14"
"HI"
"MENU"
"ARRAYS"
"ARRAYS2"
"ASSIGNMENTS"
"CASE"
"EXAM"
"FILES"
"FILES2"
"FILES3"
"FUNCTIONS"
"GRAPHICS'REVIEW"
"IF1"
"IF2"
"LOOPS"
"MOD"
"ORDER"
"OUTPUT"
"PARAMETERS"
"PROCEDURES"
"TURTLE'REVIEW"
"VARIABLES"
"---DATA FILES---"
"NUMBERFILE"
"STUDENTFILE"

**Best Of COMAL**
**Side 1**

"BOOT C64 COMAL"
"C64 COMAL 0.14"
"HI"
"-COMAL PROGRAMS-"
"APRIL'FOOL"
"ARABESQUE4"
"AUTO'DIRECTORY"
"BOUNCE'BALL"
"CHRIS'STAR"
"CLUE"
"COLOR'COMBO"
"DAISY'COLOR'DEMO"
"DIR'MANIPULATOR"
"DIR'PRINTER3"
"DISK'DATA'BASE"
"DISK'EDITOR"
"EXPLODED'PIE"
"FLURRY"
"GALACTIC'NEWS"
"GUESS'IT"
"GUTENBERG"
"HILBERT"
"INK'BLOT"
"MAGIC'FRUIT"
"MUSIC"
"POLYMUSIC"
"PRINT'ERROR'MESG"
"QUEENS"
"RANDOM'SHOW"
"SKY'CATCHER"
"SPRITE'EDITOR21"
"TURTLE'TUTOR"
"WALL'CLOCK"
"WIND'CHILL"
"-PROCS & FUNCS--"
"FILE'EXISTS.FUNC"
"SHIFT.PROC"
"---DATA FILES---"
"ALPHA2.DAT"
"COMALERRORS"
"DESIGN.DAT"
"FLAKE.DAT"
"RASTER.MEM"

**Best Of COMAL**
**Side 2**

"HI"
"---- ----------"
" MENU"
" VIEW'LIB"
" ADD'DISK"
" TEST1"
" ANNOTATE'DISK"
"----------------"
"BOOK DISKS"
"CARTRIDGE DISKS"
"TODAY DISKS"
"USER GROUP DISKS"
"oTHER"
"--------- ------"
" DBASE"

## LOAD MACHINE CODE

by Phyrne Bacon

This procedure loads machine code from disk back into its original location.

```
proc load'obj(name$) closed
 poke 858,169
 poke 859,0
 poke 860,76
 poke 861,213
 poke 862,255
 open file 78,name$+",p",read
 sys 858
 close file 78
endproc load'obj
```

IF YOUR LABEL SAYS
LAST ISSUE: 6
YOU MUST RENEW NOW.
SUBSCRIPTION CARD
IS INSIDE


# COMAL USERS GROUP USA

5501 GROVELAND, MADISON WI 53716-3251


BULK RATE
U.S.POSTAGE
PAID
MADISON, WI
PERMIT 2981

==========

IMPORTANT NOTICE

==========

YOUR SUBSCRIBER NUMBER IS PRINTED ON YOUR MAILING LABEL

PLEASE INCLUDE THIS NUMBER WITH ANY FUTURE ORDERS OR CORRESPONDENCE

THANK YOU

==========


IMPORTANT NOTICE


==========